DESIGN OF AN INTERACTIVE ACCOUNTING TUTOR

John Macko

(NASA-CR-129674) DESIGN OF AN INTERACTIVE ACCOUNTING TUTOR M.S. Thesis J. Macko (Massachusetts Inst. of Tech.) Sep. 1970 215 p CSCL 05I

N73-13192

Unclas
G3/08 49935

Sloan School of Management
Massachusetts Institute of Technology
Cambridge, Massachusetts

September 1970

DESIGN

OF

AN INTERACTIVE

ACCOUNTING TUTOR

by

John Macko

S.B. Management, M.I.T.

(1970)

S.B. Electrical Engineering, M.I.T.

(1970)

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1970

Signature of Author.........John Macko.........................
Alfred P. Sloan School of Management, August 24, 1970

Certified by..
Thesis Supervisor

Accepted by.........Abraham J. Siegel.........................
Chairman, Departmental Committee on Graduate Students

DESIGN OF AN INTERACTIVE ACCOUNTING TUTOR

John Macko

Submitted to the Alfred P. Sloan School of Management on August 24, 1970 in partial fulfillment of the requirements for the degree of Master of Science.

ABSTRACT

A project to design an interactive program to teach accounting techniques is described. The four major goals of the project are discussed and a review of the literature on teaching machines and computer-assisted-instruction is included. The system is implemented on the CTSS time-sharing system at M.I.T. and uses an ARDS graphic display. The software design of the system is described in detail. A typical session with the tutor is also described. Appendices include complete system documentation.

Thesis Supervisor: John F. Rockart
Title: Associate Professor of Management

ACKNOWLEDGEMENTS

First and foremost, I am deeply indebted to Professor Jack Rockart. Without his advice, constructive criticism, and, most especially, his sincere and continuous encouragement, this thesis would not have been possible. I am indebted to Professor Zenon S. Zannetos for the facilities and research environment which he maintains for the ALP Project.

To George Dixon and Loren Platzman, my research associates, go my sincere thanks for their insights into the many problems which arose in the development of CLOSE.

I greatly appreciate the efforts of Dearne Horn, Gail Grootemaat, Kee Ko, and Jo-Anne Sullivan who helped prepare the manuscript and diagrams.

I shall be forever indebted to Katharine H. Schwed for her cheerful encouragement throughout the entire project, and for being Kathy.

Finally, work on this project was supported by the Land Education Grant to MIT and the MIT-IBM Master Agreement for computer time.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Education is currently besieged with complex problems. It is plagued with increasing demands on an antiquated system, a shortage of well-qualified teachers, and overtaxed facilities. The past two decades have witnessed three major changes in our society which have caused these problems. One of the causes has been an exponential growth in technology heavily increasing the amount that can be learned. The rapid increase in our population and the belief that everyone is entitled to an education have also been influential factors. Taken together these three factors are largely responsible for magnifying the problem to the point that a crisis is developing in our educational systems. The situation has been compounded by the fact that education has been slow to respond to changes in technology and society. Skinner writes that "scarcely any area of human activity has been more resistant to scientific analysis and technological change than education. Although our homes, offices, factories, and means of transportation have been transformed within a generation, the typical classroom and techniques of teaching have hardly changed in a century." <42> Over the past two years the Associative Learning Project, ALP, has sought, as one goal, to discover ways to improve our learning systems. In particular, in this regard, it has established for itself the following objectives:

1) <u>To find ways to make the learning process more efficient</u>:

One of the by-products of technological progress has been an information explosion. Coupled with this increase in knowledge has been an increase in an individual's need for knowledge. Learning systems have been caught in the middle. On the one hand teaching resources, i.e. facilities and instructors' time, are limited. In addition, the amount of time and energy which a student can devote to study is also constrained.

On the other hand, there is an ever increasing demand for more knowledge. Techniques must be developed to allow an instructor to relegate the repetetive, programmed aspects of his teaching to teaching machines or other devices so that he will have more time to teach the more complex, unprogrammable topics.

Furthermore, the traditional belief that a student's time is not valuable, or at least not as valuable as the instructor's, is no longer valid. With more to learn, effective use of the student's time is necessary. In addition, changing technology, and especially its rapid rate of change, have dictated that an individual must continue to learn throughout his life. Therefore, the student may be a practicing doctor, business executive, or an engineer. Consequently, in designing learning systems for the future, the student's time cannot be discounted. Learning systems must be designed which optimize not only the instructor's time, but also the student's.

2) <u>To tailor the learning process to the student's background and interests</u>: Present educational systems tend to treat students as if they are all identical, irrespective of their individual backgrounds, abilities, needs, and interests. This is partly due to the fact that the lecture technique of teaching typically forces the instructor to gear his lecture to the average student (and sometimes to the below average student). Generally, the result is that the bright student is bored because he is not being challenged, and, at the same time, the below average

student is lost. Learning systems need to be designed which are more personalized and more responsive to the individual student.

3) <u>To integrate material across functional lines</u>: All too often a subject is taught in modular chunks, and the various pieces are never tied together. The result is that the student never gets a good overview of the whole subject. He becomes an expert in several sub-disciplines, but seldom sees how one relates to the other. And what is even worse, he seldom sees how one field relates to another field.

4) <u>To provide an experimental setting for research in the learning process</u>: Much remains to be learned about how a student learns, and about ways to optimize this process.

This paper will discuss some of the recent research which has been done in this area by the ALP project. Specifically, it will report on CLOSE, one of the systems in ALP. CLOSE is an interactive accounting tutor and is designed to assist graduate students to learn some of the basic principles and techniques of accounting.

The next section reviews the literature in the field and is followed by a discussion of CLOSE.

β

CHAPTER II

REVIEW OF TEACHING MACHINES AND CAI

A. Early Conceptual Work

S.L. Pressey <12,32,33,34> was one of the first to seriously consider the idea of using a machine to teach a subject. In the early 1920's , he designed a machine about the size of a typewriter which presented the student with multiple-choice questions. The machine was designed in such a way that it would not go on to the next question until the student had given the correct answer. The teaching program was _linear_, i.e. all students followed the same path through the subject, independent of ability or interests. Pressey in his work developed two concepts of learning which have played important roles in later teaching machines. He believed in a _law of recency_, i.e. that the answer made most recently would be the one best remembered. He felt that the learning process was not hindered by incorrect answers as long as the last response was the correct answer. Tied closely to the law of recency was the _law of frequency_. Pressey held that a student needs to be motivated throughout the program, and that one of the ways to accomplish this was to ensure that the student eventually got the right answer to each question before he could go on to the next question. Pressey writes "the correct response must almost inevitably be the most frequent, since the correct is the only response by which the learner can go on to the next question; and since whenever a wrong response is made, it must be compensated by a further correct reaction." <32> Although Pressey worked for many years on his teaching machines, they did not gain wide acceptance; it was B.F. Skinner who eventually brought the teaching machine to the classroom.

Prior to experimenting with teaching machines, Skinner <12,41,42,43> had done extensive research on animal learning behavior. He believed that he could apply the same "stimulus-response" techniques, which had been quite successful in his animal experiments, to human subjects. His teaching machines reflect this approach. Skinner developed the concept of presenting material to the student in small segments which he called <u>frames</u>. A teaching program became a particular <u>sequencing</u> of certain frames. Skinner, like Pressey, believed in the linear program, but he differed with Pressey on the nature of the student's response. Pressey favored the multiple-choice question. However, Skinner favored a fill-in-the-blank type response feeling that the very act of responding tends to cause learning; consequently, if the student was required to pick one of several possible answers, there would be a tendency for the student to remember his incorrect answers. Further, Skinner felt that recall was a much more efficient learning process than recognition.

For these reasons, Skinner's machines required the student to construct his response, and not merely recognize it as one of several possible answers. Another technique which Skinner developed was the short step. Since the very act of responding tends to cause learning, the program should be designed to minimize the number of student errors. Skinner's machines presented only a small amount of new material in each frame in an effort to ensure that most of the time the student's first answer to a question would be the correct answer. This approach also had the advantage of motivating the student through his own success.

N.A. Crowder <12,14,15,16>, the next on the scene, felt that Skinner's approach was much too rigid. He believed that the teaching machine should have the ability to adjust to each student and, therefore, developed the concept of branching within a program. Branching (also known as intrinsic programming) allowed a student to follow a path through the program that was determined by his responses to the questions. Unlike Skinner's and Pressey's machines, every student did not have to follow the same path.

Crowder's concept of the teaching machine differed greatly from Skinner's. Skinner believed in the stimulus-response approach. Crowder, on the other hand, held that the teaching machine should present material to the student and should use questions to determine if the student had mastered the concepts presented. Consequently, he felt that the overt or motor response was not fundamental to the learning process; however, it was useful as a means of providing feedback to the program.

Branching allowed the student to follow a path through the program that was dependent on his responses. The concept of adaptive or extrinsic programming extended this approach. Adaptive programming utilizes information that is external to the program such as a student's age, IQ, background in the subject, etc. Because of the complexity invoaved in meaningfully incorpprating this information into a teaching program, adaptive programming is almost always used in conjunction with a computer <15,17>. Two early examples of adaptive programming are the SAKI program developed by Gordon Pask of Great Britain and the SOCRATES program developed at the Illinois Training Laboratory <23>.

B. Two Early Experiments with CAI

The work of Pressey, Skinner, and Crowder was essentially the first phase of the research in teaching machines. Their development of basic concepts, such as sequencing, frames, and programs, paved the way for later research. In the second phase of teaching machine development, researchers became interested in trying to use the digital computer. Two experiments from this period are noteworthy.

Rath, Anderson, and Brainerd (1958) from the IBM Research Laboratories attached typewriter consoles to an IBM 650 and developed a program for teaching binary arithmetic <7,35,36,48>. This basic tutor was later expanded and courses were also offered in stenotyping, statistics, and German reading <48>.

The other experiment of note in this phase was conducted by Coulson (1962) of System Development Corporation. Coulson tied a Bendix G-15 computer to a special purpose random access slide projector. He developed the CLASS (Computer-based Laboratory for Automated School Systems) program which combined CAI with television, films, lectures, and textbooks. The system was able to handle up to 20 students at a time <7,8,9,10,19,31,47>.

C. Computer - Assisted Instruction

Late in the fifties, the computer industry became quite interested in the use of computers as teaching machines. As a result, much of the early work in CAI was done by computer-oriented people. As a rule they had been trained as engineers and not as educators or psychologists. Consequently, much of the early work tends to have a very pragmatic flavor and to emphasize applications over theory. <31>

The entrance of the computer industry into the field may be classified as the beginning of the third phase in the development of teaching machines; it also marks the beginning of a serious interest in CAI.

One of the early men in the field was Licklider (1962). Experimenting at Bolt, Beranek, and Newman with a PDP-1 to which he had attached two typewriter terminals and a cathode ray tube, Licklider sought ways to trap the student's interest. This concept of motivational trapping hinged on using the computer to constantly provide quick response and reinforcement to the student. <28>

Bitzer and Braunfeld developed PLATO (Programmed Logic for Automatic Teaching Organization) using a CDC1604. The project was begun in the early sixties and was primarily concerned with how a student reacts to interaction with a computer. The system could handle up to twenty consoles simultaneously. The system used CRT terminals which could display either computer output or slides <3,4,5,6>.

D. Recent Work in CAI

Most of the research up to this point had been on selective teaching machines. Selective teaching machines store all questions and answers in memory, and at the appropriate time output these "canned" responses to the student. A basic limitation of selective teaching machines is that the human tutor in preparing a program must try to anticipate every possible student response. If he does not, he runs the risk that the tutor will give irrelevant replies to certain student responses. One way to solve this problem is to have the tutor direct the learning process. In this way the tutor knows when to expect a student response and basically what form it will take. <47> Another approach has been to write generalized algorithms, which when given a student's response, generate either remedial material or a new question. Such programs are termed generative. Math teaching machines have quite successfully used this approach. Weizenbaum in his development of ELIZA has demonstrated that it is possible to handle verbal input as well as mathematical input using algorithms <50>. Feingold, in his PLANIT programming system, has done some work on generative CAI, but his structure is basically selective <20,21>.

Another tack recent research has taken has been to investigate the extent to which a student should be able to control the learning process. Simmons <40> has developed question-answering programs which measure a student's knowledge of a subject by semantic and syntactic analysis of his responses (questions and answers). Remedial material, if necessary, is generated. Other researchers, such as Spolsky <44>, do not feel that the question-answer technique should be left to the computer, but rather visualize a situation where the teacher is on-line and can make decisions that are difficult to program.

Grubb <24,25> of the IBM Education Research Department is a firm believer in the theory that a student learns better if he is allowed to chart his own course. His research indicates that complete student control of the program yields uniformly high post-test scores over the range of pre-test scores. However, he has found that good students do equally well under either system.

E. The ALP Project

The ALP project currently has two working CAI systems, ALP and CLOSE. ALP is designed to teach graduate students the basic principles and techniques of accounting. It is implemented on the CTSS time-sharing system at MIT and uses an ARDS graphical display (see p. 26 for a description of CTSS and ARDS). The ALP system offers the student two forms of instruction, standard programmed instruction (PI), and question-answer (QA). The PI mode presents material in an essentially linear manner with some forward branching in short steps, and requires the student tc answer questions (either multiple choice or fill-in-the-blank) which it corrects and gives reinforcement to the student. The other mode, QA, allows the student to ask the system questions on any accounting concept or term presented in the program. ALP parses the question, forms an answer based on material stored in its library, and prints it out for the student. The student can operate in either mode and can change back and forth at will. (For more information on the ALP system see <1,18,26,37,38>.)

CLOSE is an extension of the ALP system. CLOSE assumes that the student has completed the ALP program or has gained a basic knowledge of accounting principles from some other source. While ALP stresses accounting principles, the emphasis of CLOSE is on accounting techniques. CLOSE is designed to assist the student in learning how basic accounting principles are applied in practice. Since CLOSE assumes that the basic principles have been learned elsewhere, it does not teach new principles to the student, but rather serves to reinforce principles already known. CLOSE accomplishes this by allowing the student to apply his accounting knowledge to practical problems; it presents accounting problems to the student to solve and gives him feedback on the appropriateness of his solution.

F. Diimeions of CAI Systems

In summary, all CAI systems have several dimensions in common. These dimensions may be used to describe and differentiate various instructional systems. The major dimensions of all CAI systems include:

1) Sequencing: In all CAI systems (as well as programmed learning) the student follows some path through the subject. In some systems there may be one path (linear) for all students, while others allow the student to

branch according to his ability and needs.

2) Availability of extensive information: The program may or may not have information about the student (IQ, background, etc.) available to it. In general, it seems to be true that if the tutor has extrinsic information available to it and can effectively incorporate it into the learning program, it will be a more effective tutor.

3) Teaching Mode: A tutor may present material in several different ways. For example, the tutor may test or drill the student, conduct a tutorial session, or develop a socratic dialogue.

4) Structure: Most systems store all questions and answers in a sort of "canned" form, i.e. only information which has been explicitly entered by the programmer is presented; other systems have attempted to develop algorithms which generate a program's material based on the student's progress.

5) Method of response: There are two sides to this parameter-- student response and system response. The student may construct his answers or he may choose from a list. Further, he may reply in natural english or he may be required to adhere to a rigid format. Furthermore, the system may provide extensive feedback and diagnostics both for the student and for the human teacher, or it may provide little or nothing in this area (most early system fall in this category).

6) Program control: Control of the program may be handled by the system or it may be placed in the hands of the student. Traditionally, the system completely controlled the program. However, recent research indicates that there may be advantages in allowing the student to control his learning.

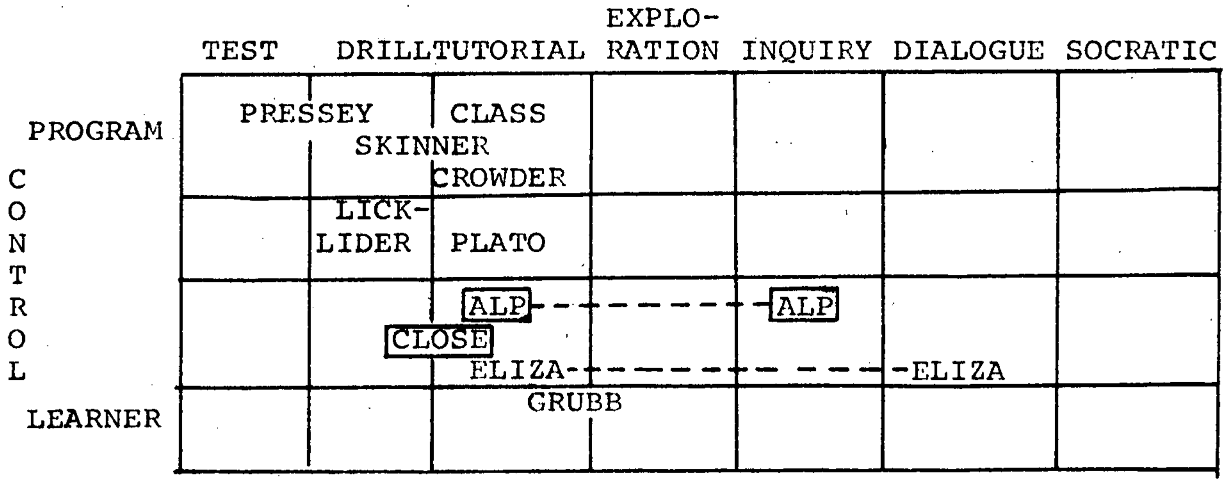
Figure 1 displays the relative positions of the programs discussed when viewed along the dimensions of sequencing and control and along the dimensions of material presented and control. (Chapter VI will discuss why CLOSE is positioned where it is in Figure 1.)

FIGURE 1.
CAI SPECTRUM

SEQUENCING

CONTROL	LINEAR		MAIN TRUNK		FULLY BRANCHED
PROGRAM	PRESSEY SKINNER		CROWDER	CLASS	
		PLATO LICKLIDER			
		$\boxed{\text{CLOSE}}$	$\boxed{\text{ALP}}$	ELIZA	
LEARNER				GRUBB	

METHOD OF PRESENTATION



CHAPTER III

CLOSE HARDWARE

A. The Time-Sharing System - CTSS

CLOSE is currently implemented on CTSS (Compatible Time-Sharing System) at MIT. CTSS is one of three time-sharing systems on campus. It is the oldest of the three, and from a technical viewpoint, is quite outdated. However, this system is still the most reliable of the three and accomodates up to thirty users on a regular basis.

The system is built around a modified IBM 7094. A core storage interval timer clock was added to the basic 7094 hardware in order to allow time-sharing. The timer is set for small bursts of time, presently 200 ms, and makes it possible to interrupt a running program to check data lines for input. Memory protection and relocation registers have also been added. These allow certain areas of core and certain instructions to be declared off-limits for the user. Communication with the system is accomplished through six data channels. Two channels interface with standard I/O equipment, i.e. printers, readers, etc. A third interfaces directly with devices that require a high-rate transfer of

c

data. The fourth and fifth interface with drum and disc storage. Drum storage, used for program swapping, consists of one IBM 7320 low-speed drum and two IBM 7320A high-speed drums. Programs are stored on an IBM 1302 disc, which has a capacity of 38 million 36 bit words. The sixth channel connects with an IBM 7750 transmission control unit and provides for communication with the timesharing consoles. Finally, the system contains two 32K core storage modules. One module, called "A core", is used exclusively by the supervisor; the other , "B core" , is used for running user programs.

The supervisor, which resides in "A core" at all times, functions as a general monitor for all operations. It is responsible for all I/O and scheduling operations. It also handles all operations involved with swapping running programs in and out of core in time-sharing the system. Such operations include program interruption, temporary memory management, and program recovery <13>.

B. The Graphics Terminal - ARDS

CLOSE uses an ARDS (Advanced Remote Display Station) graphical display. The ARDS, manufactured by Computer

Displays,Inc., is a 6 1/2" by 8" direct view storage tube (DVST). Fifty lines, with eighty characters per line, can be displayed at one time. In addition, the ARDS can display points, straight lines (both solid and dotted), and curved lines by approximating the curve with short straight line segments.

One of the basic limitations of the ARDS is that it does not have selective erasure. Because of the nature of a DVST, one section of the screen cannot be erased independently of the rest; if any part of the screen must be erased, the whole screen must be erased.

The ARDS communicates with CISS over standard voice-grade telephone lines. The interface is a type 202C data set. Currently, telephone line speeds limit the effective writing speed of the ARDS. The ARDS can draw characters at the rate of 1.2 ms per character. However, using the current 1200 bit per second line, the ARDS slows down to writing a full screen of 4000 characters in 33 seconds. In as much as the scanning speed of the typical student is well below this speed, CLOSE is not affected by this particular hardware limitation <43,44>.

CHAPTER IV

CLOSE SOFTWARE

A. Overview

CLOSE, as stated earlier, is intended to facilitate the learning of basic accounting techniques. It utilizes the programmed learning method of instruction in the belief that there is a significant section of the student population who learn material quicker and easier in such an environment. The teaching program is basically linear, but has been augmented in several respects so as to make it more responsive to the student's interests and abilities. In particular the program makes extensive use of the following previously discussed techniques:

1) <u>Student Interaction:</u> The advent of time-sharing systems has made it economically feasible for teaching machines to take advantage of the speed and data processing and storage capability of the digital computer <31>. Thus it is possible for the tutor to ask a question, wait for the student's response, analyze it, and give the student constructive feedback, in a time frame short enough such that the student

feels that he is being given individual attention. The combination of personalized instruction as well as immediate feedback makes the learning process not only more efficient, but also more enjoyable.

2) <u>Use of Graphics</u>: The truth of the old adage about the value of a picture becomes quite obvious when used in the context of the learning process. Pictures, graphs, and charts convey information in a form that the mind can more readily interpret than the same information printed in tabular form.

3) <u>Student Control:</u> As stated earlier, research <24,25> indicates that most students will learn more efficiently and have a better grasp of the subject if they are given some control over the learning process.

The general structure of CLOSE and the implementation of the above techniques in that structure are discussed in the following paragraphs.

The CLOSE system consists of five major sections. They are, in the order in which they appear in the program: introduction, posting, adjusting entries, closing entries, and financial reports. Each of these sections is described below.

The introduction welcomes the student to the system. It explains the concepts it will help the student to learn as well as the conventions it expects the student to follow when interacting with the system. When the student indicates that he understands this introductory material, the program draws a flowchart which depicts the expenditure-asset-expense cycle (see Fig. 9 on p. 61). The student sees how expenditures are acquisitions of either assets or services, and how both are eventually expensed as product or period costs. CLOSE then sets up a fictitious company GEM, Inc., and displays its "books" to the student. Currently, the "books" consist of nineteen T-accounts and their respective balances at the beginning of some chosen period. The nineteen accounts are: Cash, Inventory, Prepaid Insurance, Accounts Receivable, Accounts Payable, Notes Payable, Tax Payable, Interest Payable, Wages Payable, Retained Earnings, Capital, Wages, Advertising Expense, Interest Expense, Cost of Goods Sold, Insurance Expense, Rent, Sales, and Profit and Loss.

This set of accounts does not purport to be an all inclusive set of accounts, nor the best minimal set of accounts, but rather a set of accounts which are consistent and which allow both standard posting entries and adjusting entries to be shown.

The introductory section concludes by drawing up an initial balance sheet and income statement for GEM and displays these to the student. The student sees the contrast between the balance sheet and the income statement; he sees that the balance sheet is composed of asset and equity accounts which are permanent accounts. By contrast, the income and expense accounts, all of which are temporary accounts (and were closed out in the previous period), are shown with zero balances. The Profit and Loss Account is also a temporary account and shows the net income or expense for an accounting period. It was closed out to the Retained Earnings Account and also has a zero balance.

When the student indicates that he understands the material presented and is ready to continue, CLOSE starts to teach how to post transactions. This is the first of three major teaching sections. In subsequent sections of the program, the adjusting and closing processes are covered. In each section CLOSE presents new material in the form of a few examples and then asks the student to respond by working a few similar examples himself, with the program checking his responses and giving appropriate feedback. Specifically, in this initial posting section, CLOSE presents to the student an example transaction, i.e. the firm pays $6,000 for the use of an office and warehouse.

This transaction is posted to the ledger accounts on the screen (the Rent Account is debited $6,000, the Cash Account is credited $6,000). The postings are flagged with arrows so that the student can easily spot them (see Fig. 11 on p. 65. A console session with the CLOSE tutor, complete with diagrams, of which this example is one part is described in great detail starting on p. 59.). Another example is then displayed and posted.

At this point the program checks the student's grasp of what has been presented by giving him a transaction and asking him to post it. The student types his answer on the console keyboard. The format for his answer is relatively free as he can enter the debit and credit entries in either order, and partial answers, e.g. dr Cash 10, are acceptable. If the answer is correct, the program posts the transaction to the appropriate T-accounts. If the student's answer is incorrect, the program indicates the part(s) of the answer that is incorrect and asks the student to try again (see Fig. 11 on p. 65). Once the student has succeeded in answering the question correctly, CLOSE goes on to the next question. The program continues in this fashion, presenting transactions, asking the student to post them, and then checking his responses, for a number of transactions in order to make sure that the student has grasped the process of posting transactions.

The next stage of the program is concerned with making the end of period adjustments. An example might be to make the appropriate entries to the Inventory Account and to the Cost of Goods Sold Account, to reflect the month's sales. This part of the program uses the same technique described above to present the material. Examples are given and entries made, and then the student is asked to analyze several similar examples.

The fourth stage is concerned with closing all the income accounts, posting their balances to the Profit and Loss Account and then closing this account to the Retained Earnings Account, thereby effectively closing the firm's books for the period. Again the prodedure is as above; the program closes several accounts and posts their balances to the Profit and Loss Account, and then asks the student to do the same.

Once the books have been closed, the program summarizes in the final section the effect of all the transactions which took place during the period by drawing up a final balance sheet and income statement. The student still has before him the T-accounts -- the asset accounts with their balances brought forward, the income accounts closed out to Profit and Loss. Thus the situation is identical to that which occurred at the beginning of the program. The student is brought full circle. He sees the condition of the firm, represented by its financial statements, at the beginning of the period, the transactions which occurred, and the condition of the firm at the end of the period, i.e. the beginning of the next period. It is hoped that by presenting the material in this fashion the student will have a much clearer and broader view of the accounting process and how it affects the firm.

CLOSE endeavors to make the system responsive to the individual student's interests and abilities. It does this by making available to the student several simple commands which allow the student to tailor the program to his own needs and therefore, the particular console session. The commands are of two types: 1) technical commands and 2) teaching commands. The technical commands are COMMENT, STOP, DISPLAY, and QUESTION. These commands allow the student to correct any technical problems that may develop, i.e. accidental loss of T-account picture. The teaching commands, HELP and SKIP, allow the student to tailor the program to his needs and interests. Research (Grubb, Simmons) indicates that this approach makes the tutor more effective. The commands are described below:

1) *: CLOSE is still in the developmental stage. Comments on the system and recommendations for changes from current system users provide the system designers with essential information for system improvement. The asterisk command allows the student to type in a comment. It does not change the status of the program in any way. Comments are stored on disc where they may be examined by the system designers at a later time.

2) STOP: Situations may arise where the student does not want to complete the program. Either he may feel that he is already familiar with the subject material or time constraints may force him to quit early. The STOP command allows the student to quit the program while ensuring that all system parameters are reset. (For example, CLOSE changes the logical screen size on the display; this must be reset upon completion or early termination of the program).

3) DISPLAY: During the execution of the program, the student may inadvertently strike the erase key. This causes the screen to be erased and the T-account picture to be lost. Due to limitations in the input package, the program does not know that an erase has occurred. The program continues as if nothing

happened. In order to ensure that the system "fails soft" in such a situation, the display command was created to allow the student to redraw the T-account picture at will.

4) QUESTION: The QUESTION command was created for similiar reasons and allows the student to have a copy of the current question printed out on the screen at will.

5) HELP: One of the important functions of the human tutor is to challenge the student. If the student runs into problems, the tutor stands ready to guide him to a solution. CLOSE tries to simulate this interaction between tutor and student. At the present time the interaction between CLOSE and the student in this area is quite primitive, but the generalized structure of CLOSE allows for a substantial increase in sophistication without requiring structural changes in the system. Currently, CLOSE will recognize certain types of errors (e.g. debit and credit entries interchanged, both accounts incorrect , answer correct to a point but not complete) and output a remedial message to the student, attempting tc guide him to the correct answer. In addition, the student may at any

time type "HELP". When the student types this command, the system responds by giving him a part of the correct answer -- a part that he has not yet gotten correct. The HELP command may be called recursively. The student may type HELP, get part of the answer, type in a new answer incorporating the new information; if the answer is still incorrect, the student may get additional help by again typing HELP. By using the HELP command where needed, any student should be able to completely work his way through the program, and never get stuck on one problem.

6) SKIP: Different students have different abilities and interests. Some students will know all or most of the material. Others will not ,but will be interested only in an overview of the subject. Still others, unfamiliar with the subject will want to learn the material in as much depth as possible. And, of course, there are a wealth of students in between these extremes. One of the ways CLOSE tries to accomodate the varying needs and abilities of its students is through the SKIP command. The SKIP command allows the student to skip to the next question or to the next section (i.e. posting, adjusting, closing). If the student elects to skip to the next section, he is given

the option of seeing the questions he would have been asked, or just skipping to the next section. If he opts to see the questions he is skipping, the program presents them in a fashion identical to the way in which examples are presented. Both question and answer are given and the entries are posted.

B. Structure

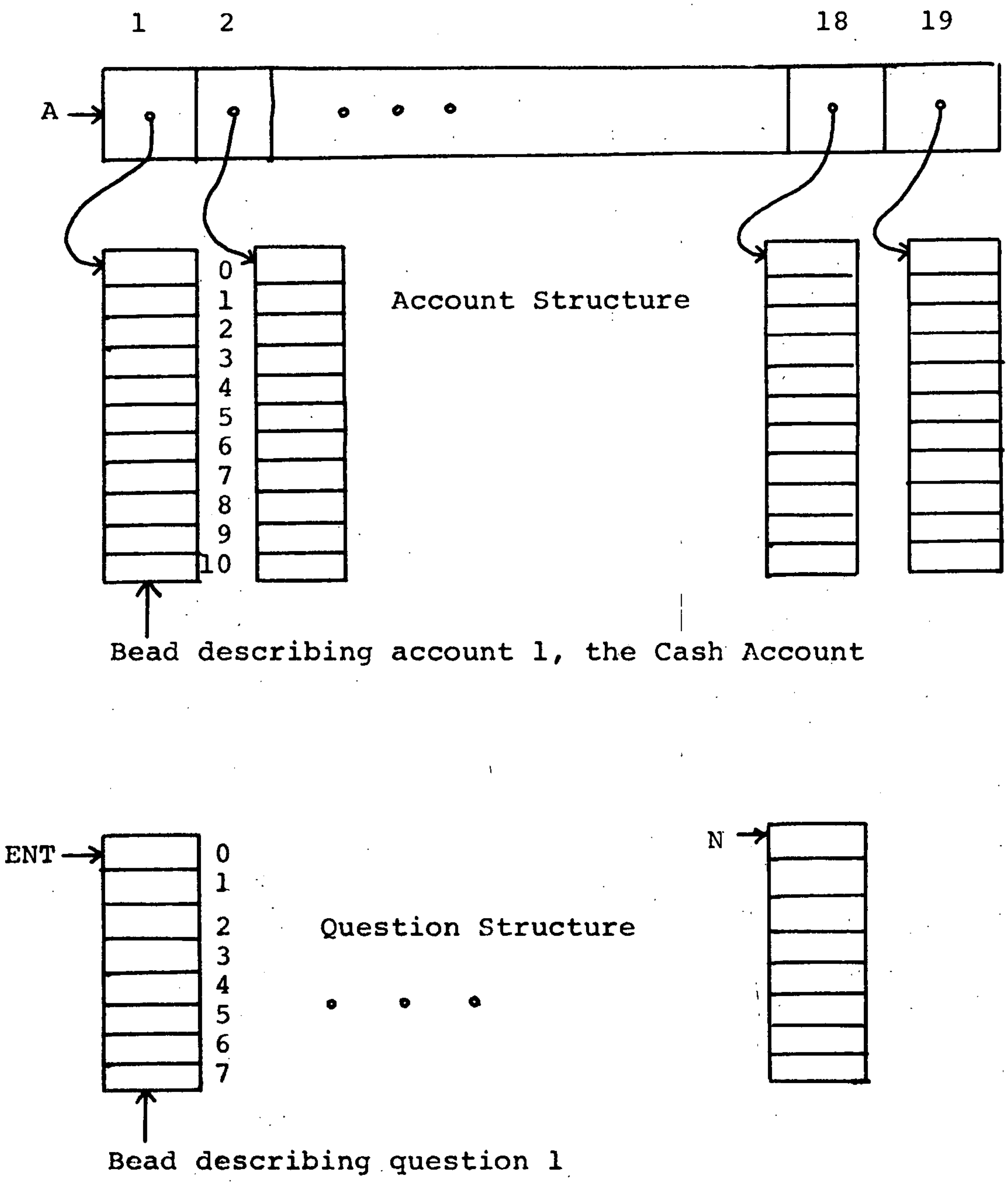
Structurally, CLOSE was designed from the outset to be a modular and relatively generalized program. Modularity was considered necessary not only to insure program efficiency, but also to make the system easily understandable by other programmers. As a consequence, the programming is straightforward and employs few coding "tricks". The structure of the program is generalized in such a manner that a range of accounting programs can be handled by the same structure. CLOSE currently has one problem set consisting of twenty examples and questions, but the generalized structure allows other problem sets to be added to the system quite easily.

CLOSE is written in AED (Algol Extended for Design), an extension of ALGOL developed at M.I.T.. AED was chosen as the language for the system for the following reasons: 1) It is a very powerful compiler-level language, allowing the user to build very complex data and list structures quite easily; and 2) CLOSE uses graphics and the ASCII character set. At the present time the best system support packages for graphics and ASCII on the ARDS are also written in AED, and communication with these routines is easiest from within AED.

CLOSE makes extensive use of the data and list structure capabilities of the AED language. All the necessary information for each T-account, e.g. X,Y-coordinates on the screen, current balance, etc., are stored as offsets of a pointer. The nineteen T-accounts pointers are stored in an array (see Fig. 2). This structure makes it possible to easily access any component of any T-account, i.e. X(A(1)) refers to the X-coordinate of the Cash Account. In a similar manner all the necessary information for each question, e.g. text of question, answer, accounts affected, etc., are also stored as offsets of pointers and the various question structures are linked together in a list structure. Consequently, all that is required to go from one question to the next is to change the pointer. A beneficial consequence of the question structure is that it is quite easy for the instructor to add, delete, or change the order of the questions.

This structure for T-accounts and questions greatly simplified the complexity of CLOSE's routines (see Fig. 3,4,5). CLOSE, the main routine, is in control at all times except when a answer is being

Figure 2.


1
2
18
19
A
0
1
2
3
4
5
6
7
8
9
10
Account Structure
Bead describing account 1, the Cash Account
ENT
0
1
2
3
4
5
6
7
Question Structure
N
Bead describing question 1

checked; then CHECK, described later, is in control. CLOSE is composed of two sections -- initialization and question-answer. During initialization CLOSE makes subroutine calls which set up the T-account and question structure for the problem set requested (currently only one problem set); it also opens disc files used by the program to record student answers, initializes system parameters, and calls routines which print the system introduction and record the time, date and name of the student.

The question-answer section is a very simple loop: (see Fig. 4 on P. 44. Each of the dotted line rectangles represent one of the following steps.)

1) Print question.

2) CHECK answer, if the question is to be ansered by the student, and print feedback; otherwise go to 4.

3) if the answer is correct, go to 4; otherwise, wait for new answer and go to 2.

4) Print answer, post entries.

5) If there are more questions, check remaining space on screen (erase if necessary) and gc to 1; otherwise, stop (see Fig. 6).

Figure 3.

MACRO STRUCTURE OF CLOSE

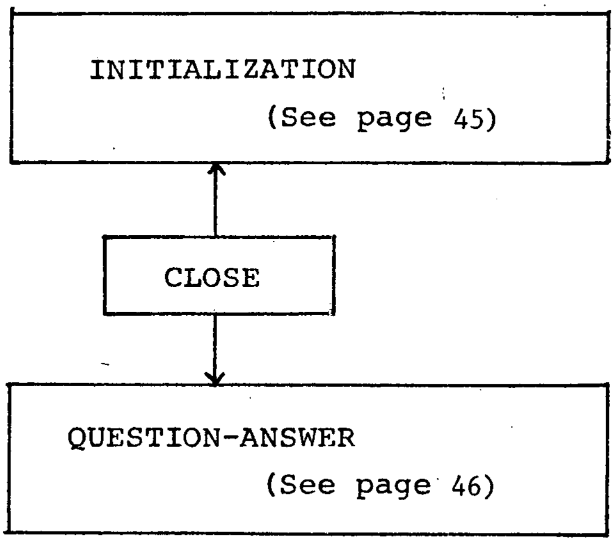


Figure 4.

MACRO STRUCTURE OF CLOSE - INITIALIZATION

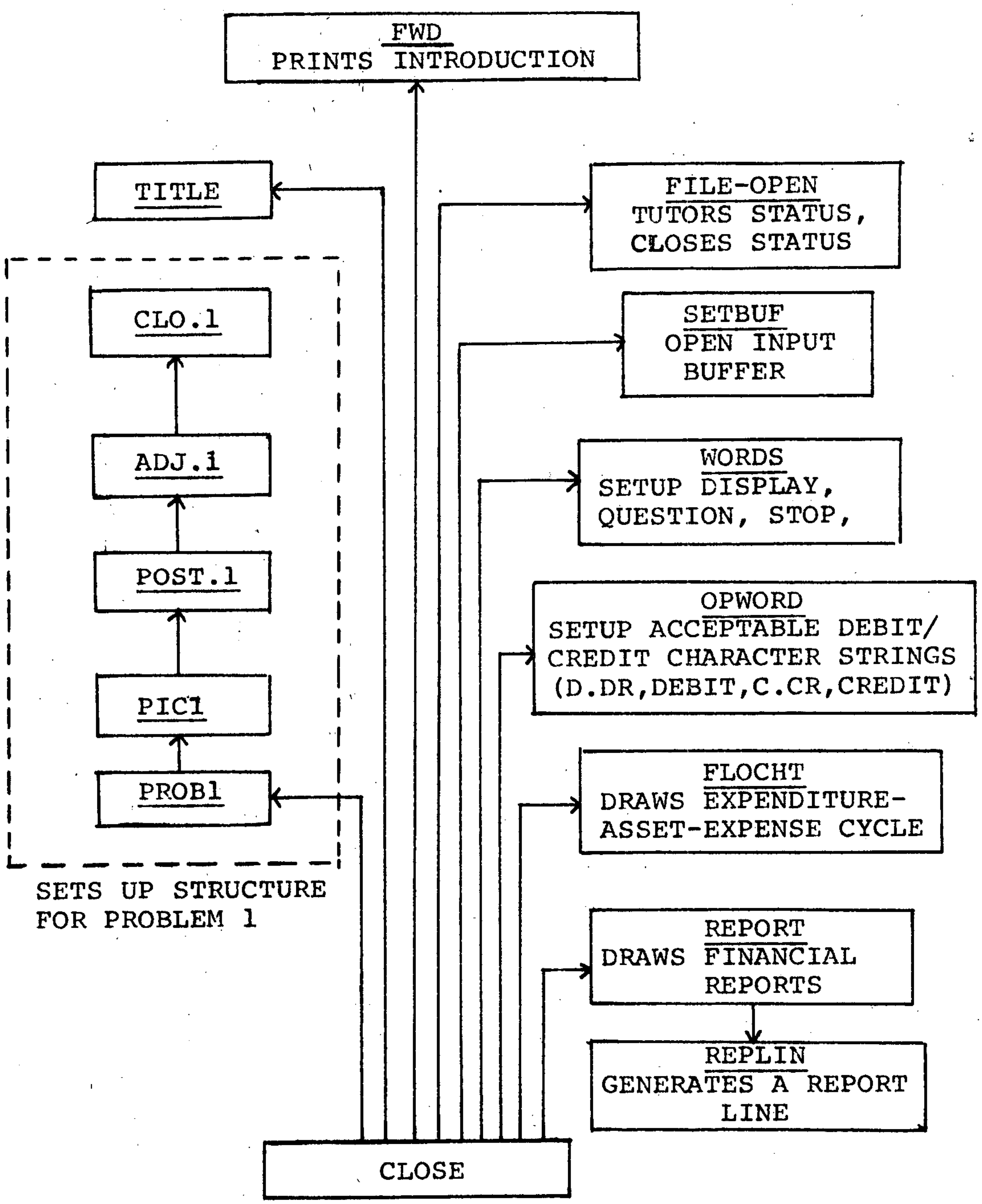


Figure 5.

MACRO STRUCTURE OF CLOSE - QUESTION-ANSWER

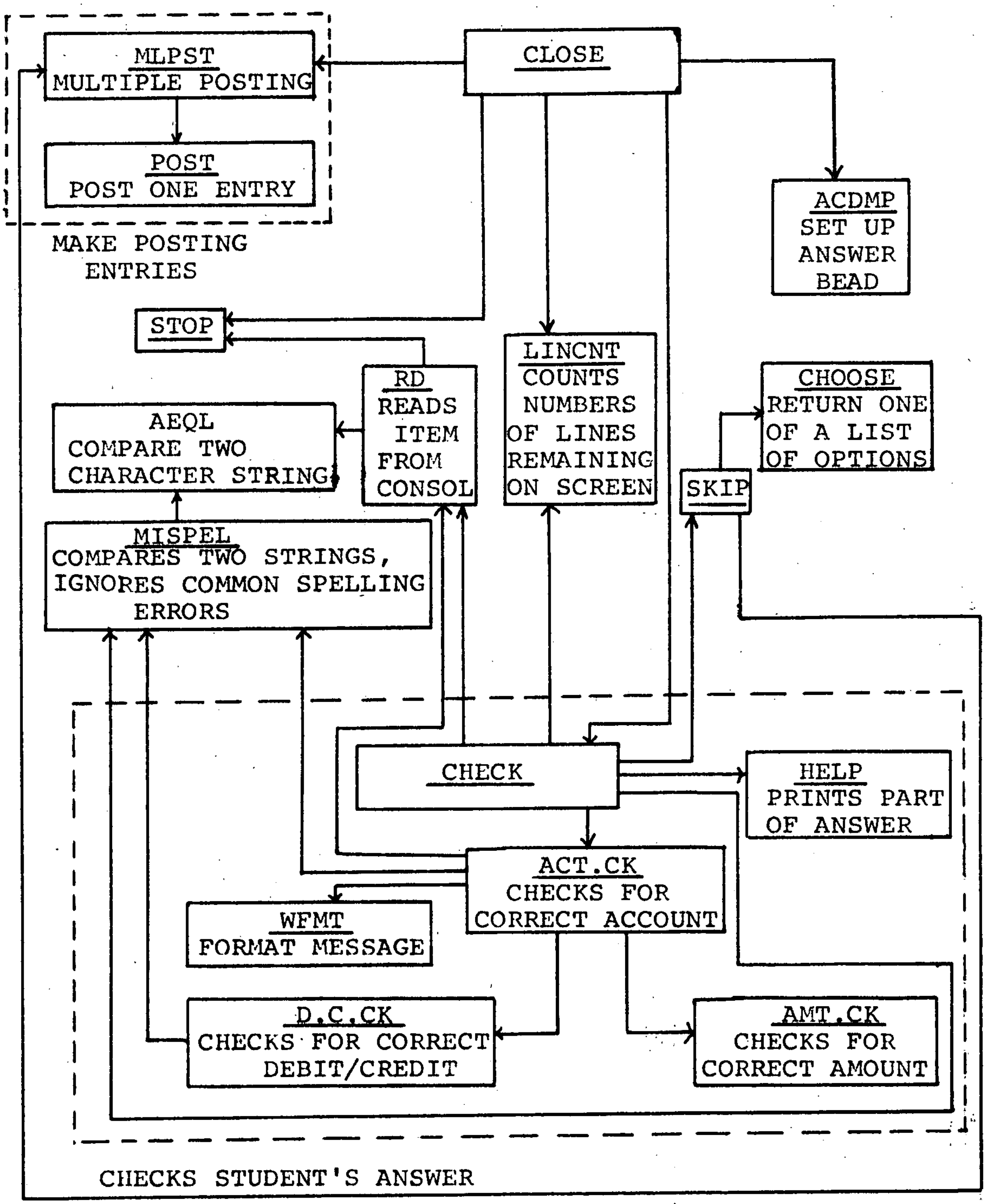
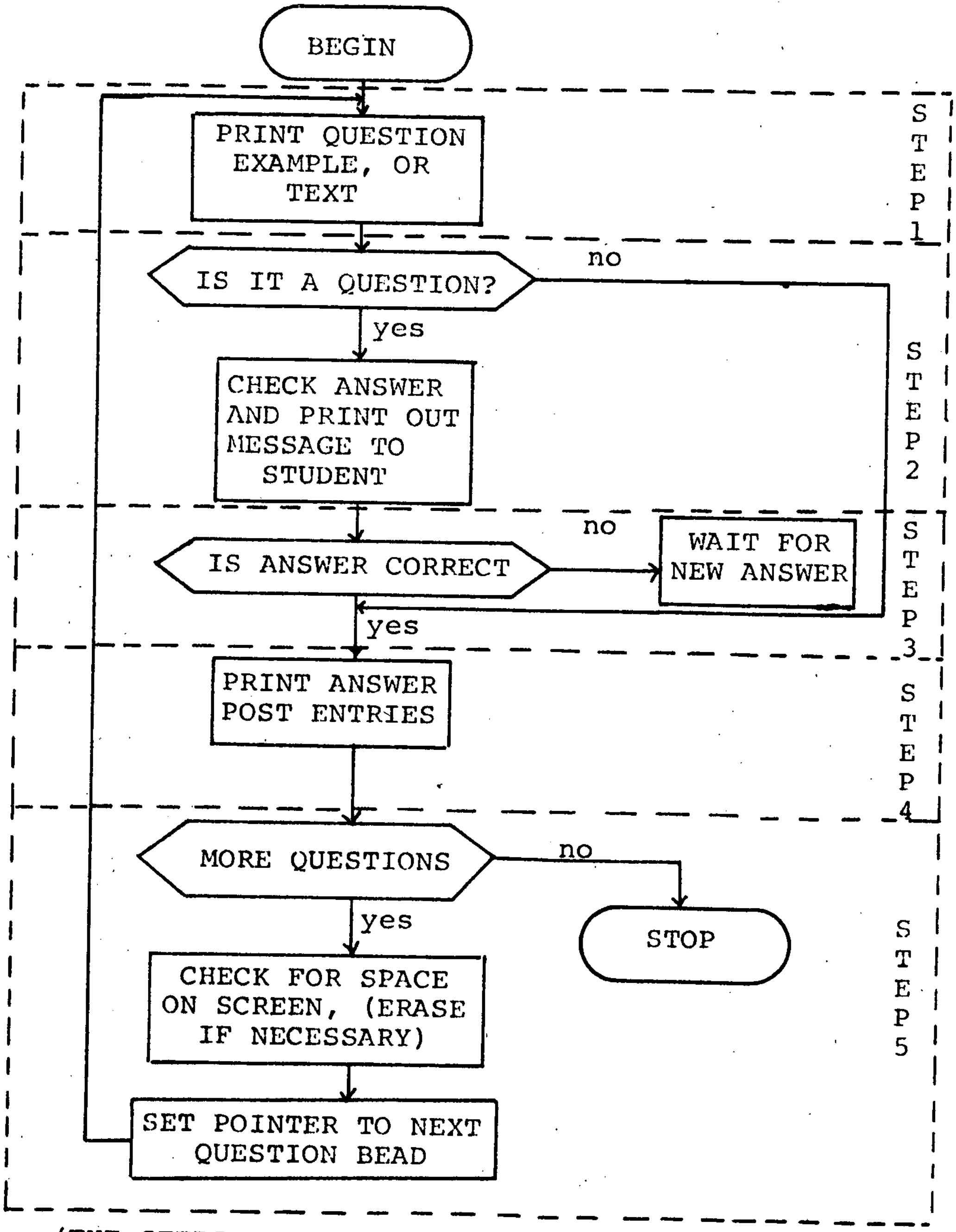


Figure 6.

QUESTION ANSWER LOOP



(THE STEPS REFER TO THE TEXT OF PAGE

As can be seen, the basic structure is quite simple. However, some quite complex issues underlie this simple structure. There are two major issues which deserve further discussion: 1) Input/Output -- how the student interacts with the system, and 2) the parsing algorithms which check the student's answers. Both of these issues are discussed below.

C. System Input/Output

The obvious issues here are: 1) how does a student enter his answers, and 2) how does the system respond.

At the present time there are basically only two types of input devices available on the ARCS and both were considered. The first is the mouse or joystick. With this device the student can position the beam on the screen, and by pressing buttons cause the location of the beam to be recognized by the system. This capability seemed a natural for the CLOSE application.

The program was originally designed in such a way that when the program requests an answer, the student responds by positioning the beam at the letters "dr" on the screen, then at the account debited, and finally at the amount. This procedure would be repeated for the credit account. Amounts, in the original design, were entered by using the following technique. Display the integers 0-9 on the screen as shown below:

dr cr 0 1 2 3 4 5 6 7 8 9

The student could enter a number by successively pointing at the appropriate integers, e.g. 125 would be entered by pointing to 1, then to 2, and then to 5.

This approach had the advantage that it partially overcame the ARDS limitation of non-selective erasure. By allowing the student to point to objects on the screen, the amount of text that had to be displayed on the screen was minimized, and hence the total number of times that the screen had to be erased was minimized.

This particular approach was thoroughly tested. The approach seemed feasible, but because of the way the AED support packages are designed, major difficulties were uncovered. First of all, the procedure followed by the student for each transaction turned out to be fairly complex and quite lengthy. Secondly, transactions requiring more than two entries were difficult to program. Consequently, this method of inputting an answer was rejected.

An alternate approach, the one finally adopted, is to have the student type his answers directly on the keyboard console. This has the advantage of being a simple operation as well as being similar to the actual process of entering transactions in a journal. However, this approach tends to fill up the screen much more rapidly, hence requiring more erasures. Upon implementation, this problem of filling up the screen was not found to be a major problem, at least at the present time. Currently, the program has twenty transactions which, on the average, require five erasures. The delays involved have not been found to be an inconvenience. Therefore, the typewriter has been adopted as the input device.

Once it was decided how the student would input his answers, the question of how the program would handle output had to be answered. In an earlier program, LEDGER <27>, all output was handled using AED graphics system support routines which actually positioned the beam on the screen at the desired location and then printed the output. This approach could have been used in CLOSE, but because of the problems encountered in LEDGER in determining exactly where to output a line (when a student can make any number of errors), it was decided to develop a fresh approach.

CTSS has a similar problem in that the system must always be able to determine where to position its next output line. Routines have been written which determine the next output line. CTSS automatically prints output on the next available line; therefore, if CTSS routines are used, there is no problem determining where to output the next question, i.e. right after the last response, no matter how many there are. However, CTSS always starts printing at the top of the screen and continues until it reaches the bottom. This would have caused answers and text to be written over the T-accounts appearing at the top of the screen. A method had to be found to keep CTSS from overwriting the accounts. The CTSS command SETPRM allows the programmer to alter the logical screen size. By altering the screen's parameters, CTSS can be made to believe that the top of the screen begins <u>anywhere</u> on the screen. With the use of SETPRM the problem was solved. System support routines were used to output the T-accounts and their entries since it was necessary to plot these at specific locations on the screen. For the remaining output, regular CTSS output (coupled with special routines to handle ASCII as CTSS normally outputs only BCD) was used with the reduced logical screen.

D. Parsing Algorithms

CLOSE expects the CHECK routine to perform two important functions: 1) it must determine whether the student's answer is correct or not and print out appropriate feedback; and 2) it must handle any commands (QUESTION, DISPLAY, SKIP, etc.) in the same manner as CLOSE does at the top level.

This last function, with the exception of SKIP, is relatively simple. RD, the routine which reads input from the console, is used in both CHECK and CLOSE. RD will recognize and appropriately handle QUESTION, DISPLAY, STOP, and * wherever found. SKIP and HELP are special cases. HELP is meaningful only when called from within CHECK, i.e. a question has been asked and the answer is being checked. HELP is ignored if encountered within CLOSE. SKIP, on the other hand, is meaningful both in CLOSE and CHECK. When encountered in CLOSE, the program can start SKIPing immediately. However, when encountered in CHECK, the current question must be answered and a return to CLOSE initiated after setting a flag which informs CLOSE that the student typed SKIP.

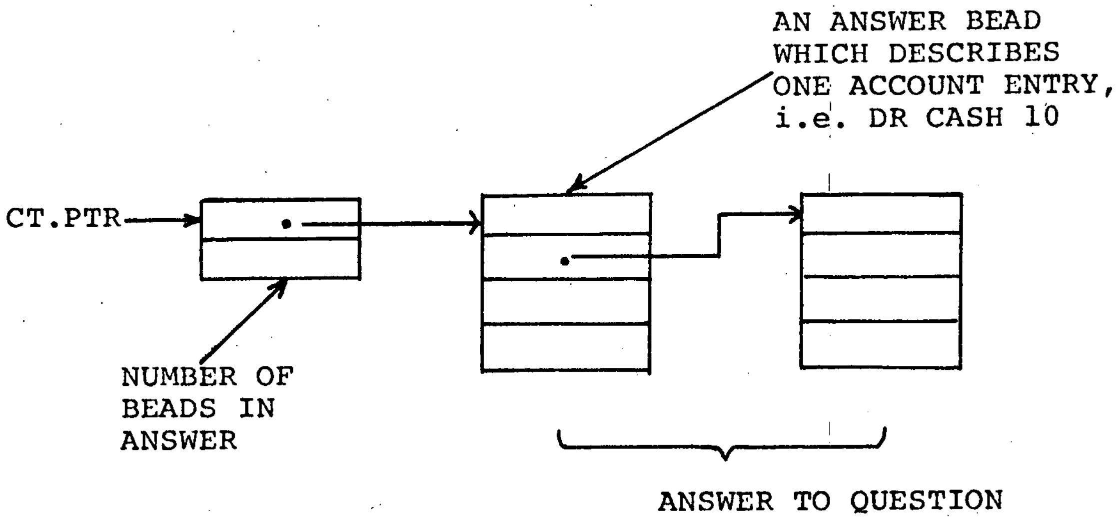
CHECK's other function, to determine if the answer is correct, is more complex. CLOSE, when it calls CHECK, passes as an argument a pointer to the correct answer. This pointer is called ACT.PTR (see Fig. zzz). CLOSE also frees a null pointer called RT.PTR. As parts of the answer are found to be correct, they are moved from ACT.PTR to RT.PTR. When ACT.PTR is null, the student has given the correct answer.

In checking the parts of an answer, CHECK takes advantage of the natural structure that exists in ledger entries. All entries are of the form <debit or credit> <account> <amount> <debit or credit> <account> <amount> ... Each entry is composed of one or more debit entries and one or more credit entries. Each debit or credit entry is composed of three items: 1) the words debit or credit (or some abbreviation thereof), 2) the account name, and 3) the amount. CHECK, therefore, checks each group of three items against the answers pointed to in ACT.PTR. If CHECK does not recognize either debit or credit (or some abbreviation) as the first item of each three item group, it flags a format error and informs the student that the format of his answer is incorrect.

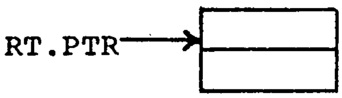
If CHECK can, it forms an acceptable three item group from the student's input, then uses ACT.CK and AMT.CK to see if the debit or credit account name entry is correct. ACT.CK looks through the accounts specified in ACT.PTR. It also checks the accounts in RT.PTR as the part of the answer being checked may have been moved to RT.PTR by a previous

Figure 7.

ACT.PTR AND RT.PTR



AN ANSWER IS A LIST OF ANSWER BEADS.
THE POINTER TO THE LIST IS ACT.PTR



RT.PTR IS A POINTER TO THE LIST OF ANSWER BEADS WHICH THE STUDENT HAS ENTERED CORRECTLY, AN ANSWER BEAD IS MOVED FROM ACT.PTR TO RT.PTR WHEN THE STUDENT TYPES AN ANSWER WHICH MATCHES AN ANSWER BEAD.

partially correct answer. It attempts to match the account name of the three item group with an account name in either ACT.PTR or RT.PTR. An exact match is not required as CHECK utilizes a misspelling algorithm to cancel the effect of spelling errors.

The misspelling algorithm is a modification of the one designed by Anderson <1>. The algorithm reports a match between two character strings if: 1) the first two and last characters of the strings match, or 2) a substring of the characters match. The length of the substring required for a match is a variable set by the programmer. Currently, a substring of length four must match.

If ACT.CK finds the account name in either ACT.PTR or RT.PTR, it checks to see if the account is being correctly debited or credited. It also checks to see if the amount is correct. AMT.CK performs this function. If a three item group is found to be correct, it is moved from ACT.PTR to RT.PTR and the next group is checked.

If a group is found to have an error, i.e. wrong account, incorrect debit or credit, or wrong amount, the incorrect item(s) of the group is surrounded by asterisks and a flag is set indicating the type of error found.

When all groups have been checked, i.e. the student's answer has been completely parsed as signified by the discovery of a carriage return, the error flags are examined. If any of the flags have been set, the student is told that his answer is incorrect, and his answer, with the incorrect item(s) surrounded by asterisks, is printed out (see Fig. 11 on 65). If no flags have been set, the student is informed that his answer is correct. In either case CHECK returns to CLOSE. If the answer was incorrect, CLOSE calls CHECK to examine the next answer. Otherwise, CLOSE prints out the answer, posts the entries, and goes on to the next question.

CHECK currently has six error flags. They appear as a six digit code and are stored on disc alongside each student response. The code d0 d1 d2 d3 d4 d5 is as follows:

d0	total number of errors
d1	" " " debit errors
d2	" " " credit errors
d3	" " " account errors
d4	" " " amount errors
d5	" " " format errors

By performing logical operations, such as AND and OR, on the error codes, useful diagnostics can be established.

Currently, CHECK recognizes and gives a special error message to the student for each of the following errors: 1) answer completely incorrect, 2) debit and credit entries reversed, and 3) answer correct, but not complete. Other logical combinations can be used to detect other types of errors.

If the student's answer is completely incorrect, CHECK informs the student that he may get help with the problem by typing HELP. HELP pulls the first account off the ACT.PTR list and prints it out as part of an appropriate message, i.e. Cash is one of the accounts, or Cash is the other account. As stated earlier, HELP may be called at any time after a question has been asked, and it may be called recursively.

E. Other Routines

There are several other routines which play important roles in CLOSE. Examples are AEQL which compares two character strings, or CPY.LN which empties the read buffer and creates a pointer to a copy of it. As the techniques employed are rather well known, these routines are not discussed here. The interested reader is referred to the system documentation for a complete description of each routine.

CHAPTER V

A SESSION WITH CLOSE

This section describes in detail how a student interacts with the CLOSE system. The diagrams show what appears on the screen, while the text describes the session. Although not every display appears as a diagram, the diagrams are representative of the type of displays in the CLOSE system.

CLOSE starts the session of by welcoming the student, (in this session a young lady named Kathy), to the system. It then prints out an introduction <see Fig. 8> which describes the goals of the system as well as the conventions the system expects the student to follow.

When Kathy has read and understood the instructions, she hits the erase key. Her action erases the screen and informs CLOSE that the student is ready to continue. CLOSE then requests and records the student's name.

Fig. 9 shows the next display. Here the student is graphically shown the expenditure-asset-expense cycle. Again, when Kathy is ready to continue, she strikes the erase key.

The next display <see Fig. 10> presents the student with the "books" of the company, a short description of the

CLOSE is a program written in AED designed to take advantage of the interactive capabilities of a computer, and to put these capabilities to good use as a working tool for the teaching of accounting.

Specifically, CLOSE is designed to help you learn the following three aspects of accounting: 1) the process of posting transactions to a ledger,
2) making adjustments to the ledger accounts at the end of a period,
and 3) closing out the ledger accounts.

CLOSE employs the technique of presenting several example transactions and then asking you to give the answers to several similar transactions. The program assumes that you will abide by certain conventions; these conventions are listed below:

1) When "READY" appears on the screen, it means that the program is waiting for you. When you are ready to continue, and have typed a response, if one was expected, you should hit "new line".

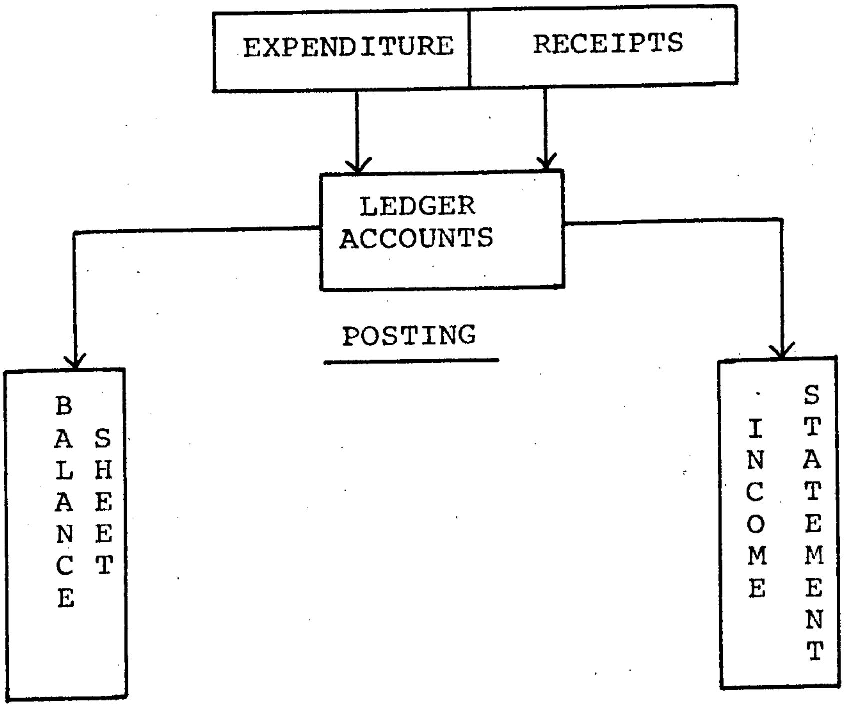
2) Example transactions are denoted by the prefix "EXAMPLE". You are not expected to answer these, but you should make sure that you understand each one before you continue with the program.

3) Transactions which expect you to respond are prefixed by "QUESTION". The format of a response is similar to that of a journal entry, i.e. an account and the amount to be debited is specified, followed by the account and the amount to be credited. Specifically, you respond to a question by typing "dr" followed by the account to be debited, followed by the amount (in thousands of dollars); on the same line you then type "cr" followed by the account and the amount to be credited. Then hit "new line" to continue.

When you have read and understood the above directions, hit "erase".

Figure 9.

EXPENDITURE-ASSET-EXPENSE CYCLE



Accounting is concerned with recording changes in the firm's assets and liabilities. To a large extent, these changes are day-to-day-expenditures and receipts of cash.

READY (LF)

As they occur, these day-to-day changes are recorded in the firms "books" which are called its ledger accounts.

READY (LF)

If the change is concerned with the operation of the business during the current period, the change is called an expense (or income) and at the end of the period, is recorded in the financial record of the current period -- the income statement.

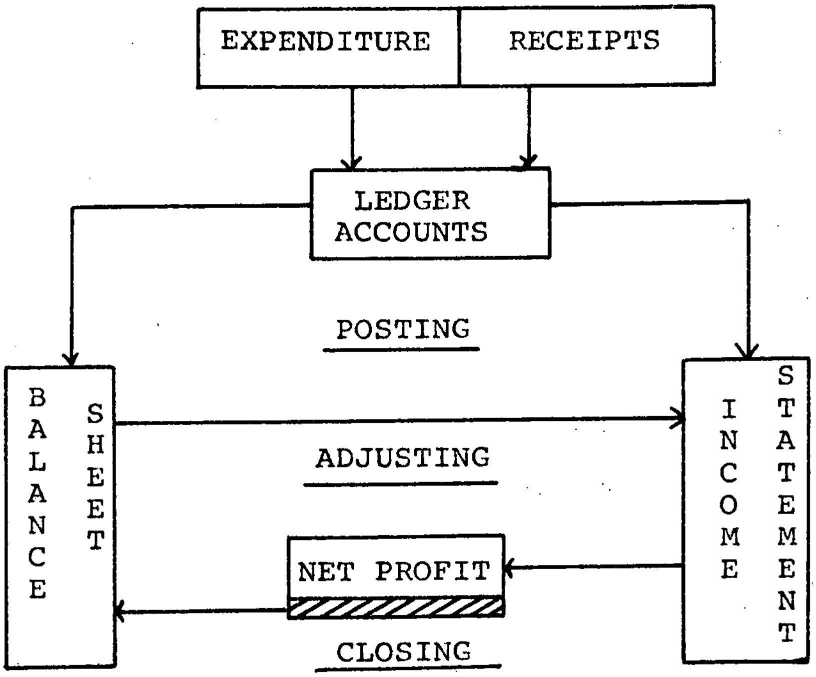
READY (LF)

However, some expenditures are made to acquire assets which will have value remaining at the end of the current period (An example is a building). These expenditures are usually posted to the balance sheet to record the continuing value of the assets to the firm. <u>This is the posting process</u>.

READY (LF)

Figure 9 continued

EXPENDITURE-ASSET-EXPENSE CYCLE



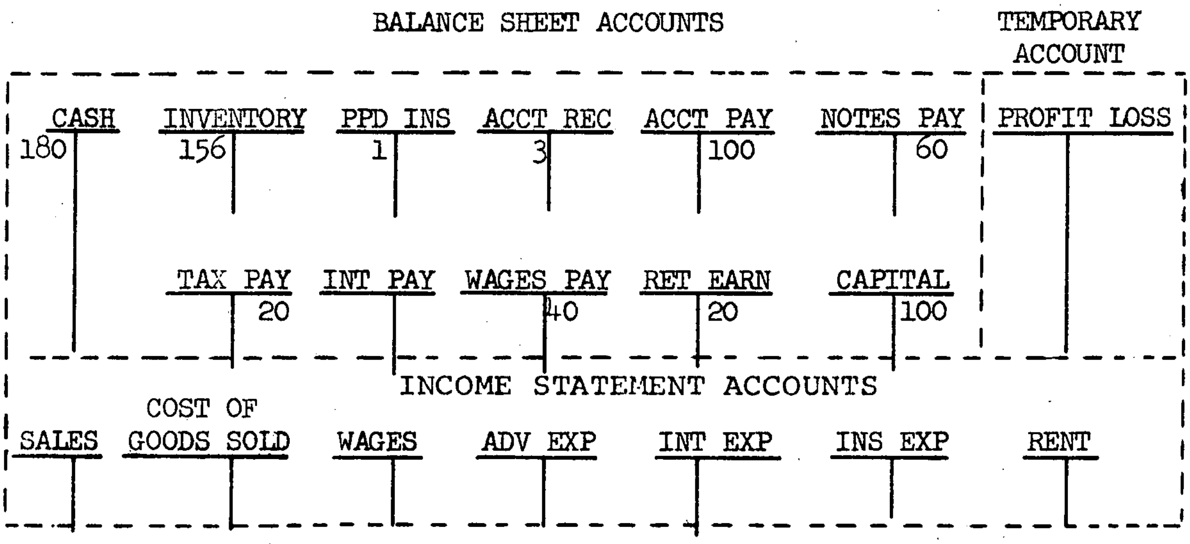
At the end of a period, some of the balance sheet assets will have contributed some of their value (i.e., depreciation of a building) to the current period's business effort and this value reduction must be recorded to "this period's" expenses. In like manner, liabilities of the firm carried on the balance sheet may become income of the firm during the period (e.g. prepaid rent). In addition, in some non-manufacturing firms, the inventory which has been sold will be subtracted from the balance sheet at this time. <u>This is the prcess of adjustment</u>.

READY (LF)

The difference between the income and expenses for a period represents the net change in value of the firm during the period. To determine this net change for the period, all income and expense accounts are closed to a temporary account called the profit and loss account. The balance of the profit and loss account represents the net change, or net profit, for the period. As the net profit is an increase (or decrease) in the value of the firm, it is posted to the balance sheet. <u>This is known as the closing process</u>.

READY (LF)

Figure 10



These accounts are the "books" of GEM, Inc. GEM is engaged in the manufacture and sale of Adzes, an esoteric plant made from a hemp plant and widely held to be the cure for all of man's ills.

You have been hired as an accountant and are responsible for keeping GEM'S books.

BALANCE SHEET

ASSETS		
cash	180	
inventory	156	
ppd ins	1	
acct rec	3	
TOTAL ASSETS		340
EQUITIES		
LIABILITIES		
acct pay	100	
notes pay	60	
tax pay	20	
int pay	0	
wages pay	40	
TOTAL LIABILITIES		220
ret earn	100	
capital	20	
TOTAL EQUITIES		120
		340

INCOME STATEMENT

sales	0	
cost of goods sold	0	
GROSS PROFIT		0
OPERATING EXPENSES		
wages	0	
adv exp	0	
int exp	0	
ins exp	0	
rent	0	
TOTAL OPERATING EXPENSES		0
NET PROFIT		0

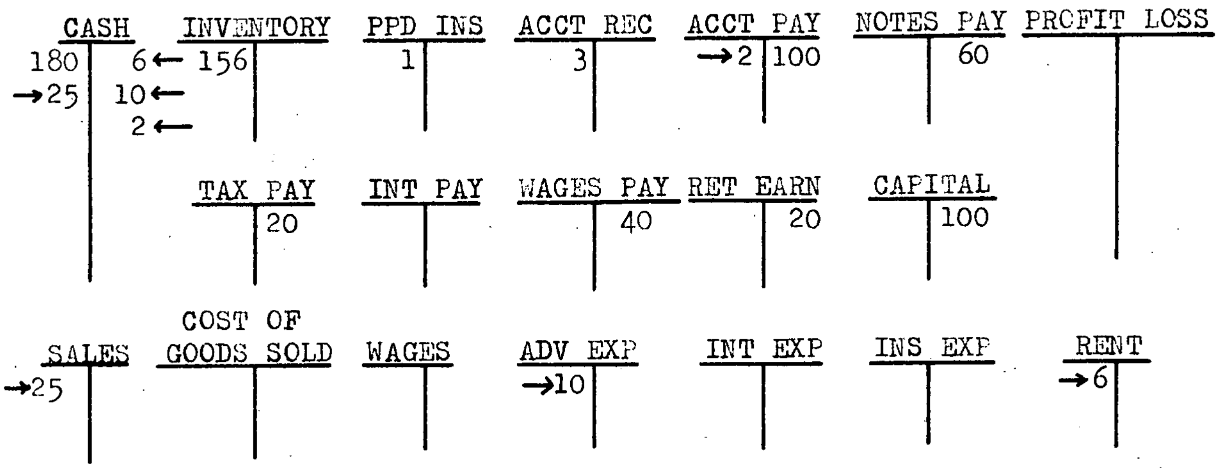
company, and an initial balance sheet and income statement.

Upon a signal from the student that she is ready to continue, CLOSE erases the screen and draws the display shown in Fig. 11. At this time the T-accounts and Example 1, transaction (1) in the diagram, are on the screen. The lower part of the screen is blank. Since (1) is an example, CLOSE immediately prints out the answer <see (2)>. The answer is also posted to the accounts on the screen. Note the "6<-" credit entry in the CASH account and the "->6" debit entry in the RENT account. The arrows help the student to spot the posting. CLOSE then prints "READY" and waits for the student. Kathy, after examining the transaction, strikes new line (LF) to tell CLOSE to continue.

Example 2 is then presented <see (3)>, the answer is printed on the screen, and the entries are posted. A new line from the student causes the program to continue.

CLOSE then asks a question -- how should the advertising fee be posted <see (4)>. Kathy types "dr adv exp 10 cr acct pay 10 (LF)" which is incorrect. CLOSE responds with "Your response of dr adv exp 10 cr **acct pay** 10 is incorrect, try again." Kathy sees her mistake and types in "cr cash 10 (LF)" to correct the answer. CLOSE responds "Your answer is correct, Kathy." It also prints

Figure 11



(1) <u>EXAMPLE 1</u> : You pay $6,000 for the use of an office and a warehouse. What ledger entries should be made ?

(2) <u>ANSWER</u> : dr rent 6 cr cash 6 (LF)
READY (LF)

(3) <u>EXAMPLE 2</u> : GEM, Inc., just sold $25,000 worth of its Adzes for cash. What should be posted ?
<u>ANSWER</u> : dr cash 25 cr sales 25 (LF)
READY (LF)

(4) <u>QUESTION 1</u> : GEM engages CON, Inc. to handle its advertising. You pay their feee of $10,000. Enter the transaction.
READY dr adv exp 10 cr acct pay 10 (LF)
Your response of dr adv exp 10 cr **acct pay** 10 is incorrect, try again.
READY cr cash 10 (LF)
Your answer is correct, Kathy.
<u>ANSWER</u> : dr adv exp 10 cr cash 10
READY (LF)

(5) <u>QUESTION 2</u>: You pay a bill of $2,000 for equipment bought on account in the previous period.
READY dr cash 2,000 cr acct pay2,000 (LF).
Your response of **dr** cash 2000 **cr** acct pay 2000 has the debit and credit entries interchanged, Kathy, try again.
READY dr acct pay 2 cr cash 2 (LF)
Your answer is correct, Kathy.
<u>ANSWER</u> : dr acct pay 2 cr cash 2
READY (LF)
READY [TO ERASE, HIT NEW LINE] (LF)

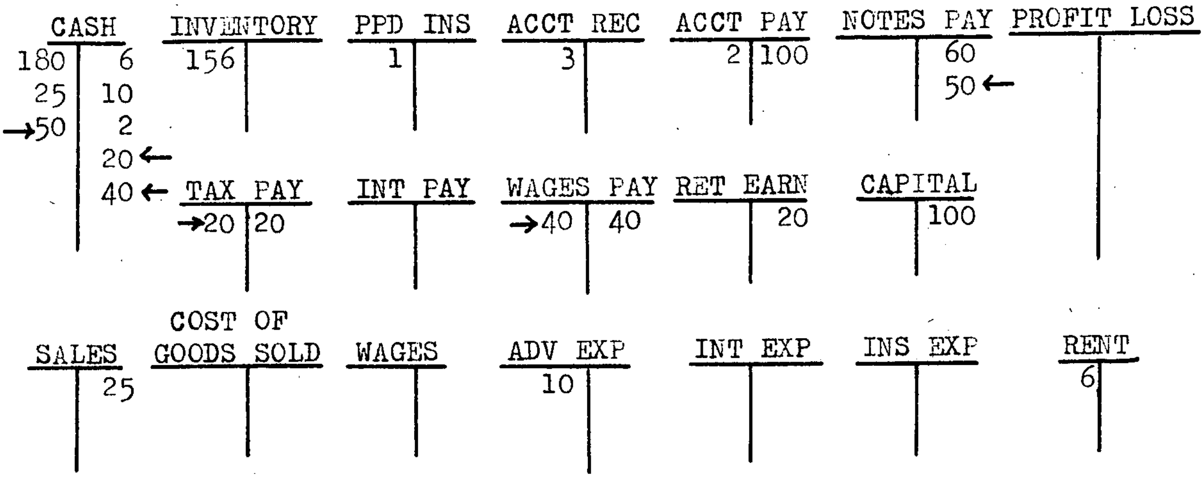
out a copy of the correct answer, posts the answer to the T-accounts involved, and waits for the student.

Kathy types new line and CLOSE prints out the next question <see (5)>. Here, Kathy responds by typing "dr cash 2,000 cr acct pay 2,000 (LF)". Note the use of commas in the amount as well as the fact that the amount is typed in full. In this case Kathy has her debit and credit entries reversed. CLOSE responds "Your response of **dr** cash 2000 **cr** acct pay 2000 has the debit and credit entries interchanged, Kathy, try again." Kathy reverses her entries and types in the correct answer. CLOSE again acknowldges that her answer is correct, and prints and posts the answer. When Kathy strikes new line (LF) this time, CLOSE responds "READY (TO ERASE, HIT NEW LINE)". This informs Kathy that when she strikes new line, the screen will be erased. Kathy looks over the entries that have been made, and strikes new line.

The screen is erased and the T-accounts are redrawn <see Fig. 12>. Note that the arrows pointing to earlier entries have disappeared.

CLOSE now asks a question about posting a bank loan. Kathy responds correctly, but mispells NOTES PAY. She types in "nots pay" instead of "notes pay" <see (6)>. CLOSE checks the spelling, decides that she means "notes pay", and

Figure 12



(6) QUESTION 3 : GEM borrows $50,000 from the bank.
READY cr nots pay 50 dr cash 50 (LF)
Your answer is correct, Kathy.
ANSWER : dr cash 50 cr notes pay 50
READY (LF)

(7) QUESTION 4: GEM pays $20,000 for taxes that were due in the previous period.
READY dr tax pay 20 cr cash 20 (LF)
Your answer is correct, Kathy.
ANSWER : dr tax pay 20 cr cash 20
READY (LF)

(8) QUESTION 5 : GEM pays $40,000 in wages that were due at the end of the last period.
READY skip (LF)
ANSWER : dr wages pay 40 cr cash 40
(A) Print questions and answer so I know what I am skipping.
(B) Skip to the next section.
(C) Skip to the next question.

Type desired option : c (LF)

READY [TO ERASE, HIT NEW LINE] (LF)

responds:

> "Your answer is correct, Kathy."
>
> "<u>ANSWER</u>: dr cash 50 cr notes pay 50"

Kathy answers the next question correctly, and when CLOSE presents QUESTION 5, she decides that she wants to "skip" it. She types "skip" and CLOSE responds by giving the answer to QUESTION 5 and also a list of skipping options <see (8)>. Kathy decides to skip only this question and types in a "c". However, the screen is full, and CLOSE waits for a new line before erasing it.

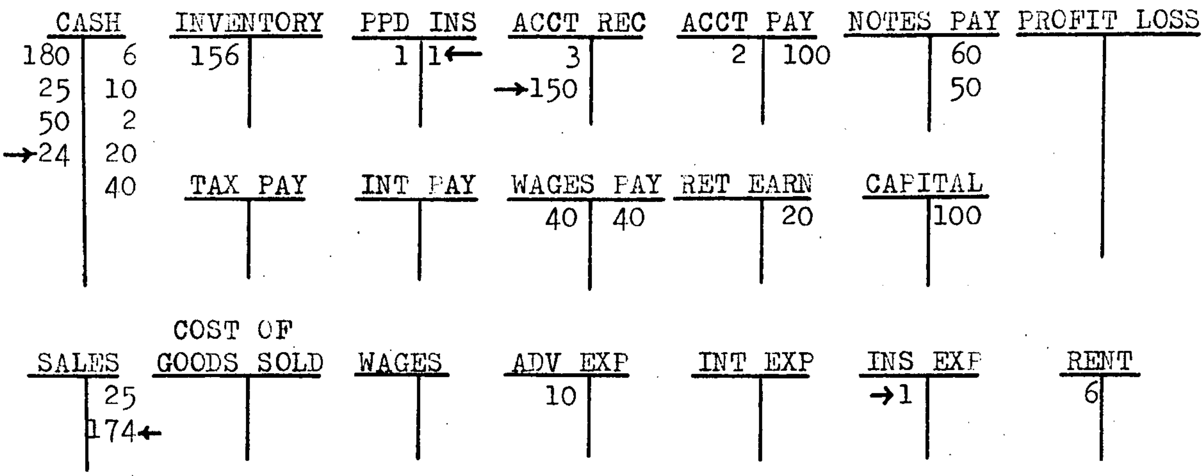
After the T-accounts are redrawn, CLOSE presents QUESTION 6 <see (9)> in Fig. 13. Kathy decides to answer a part of it at a time as she is not sure of the complete answer. She starts with the Cash entry. CLOSE guides the way by responding that her answer is correct so far, but that there are more entries. She continues in this fashion until she has the answer completely correct.

When Kathy types new line, CLOSE starts the adjusting entries section <see (10)>. Kathy types in a comment that the program should have more posting questions, and CLOSE thanks her for her comment and records it on disc.

Kathy then decides to experiment with the "skip 1" command. The optional argument "1" causes a skip to the

f

Figure 13



(9) QUESTION 6 : GEM sells $174,000 worth of its Adzes. The buyer pays $24,000 cash and will pay the balance in 30 days.
READY dr cash 24 (LF)
Your answer, dr cash 24, is correct so far, type in the other entries.
READY dr acct rec 150 (LF)
Your answer, dr acct rec 150, is correct so far, type in the other entry.
READY cr sales 174 (LF)
Your answer is correct, Kathy.
ANSWER : dr cash 24 dr acct rec 150 cr sales 174.
READY (LF)

(10) Some events which affect the T-accounts do not generate journal entries. Adjusting entries must be made at the end of the month to account for these events.
READY *I think the program should have more posting entries (LF)
Thank you for your comment.
READY skip 1 (LF)

(11) EXAMPLE 1 : At the start of the year a $12,000 insurance premium was paid record this month's expense.
ANSWER : dr ins exp 1 cr ppd ins 1
READY (LF)

READY [TO ERASE, HIT NEW LINE] (LF)

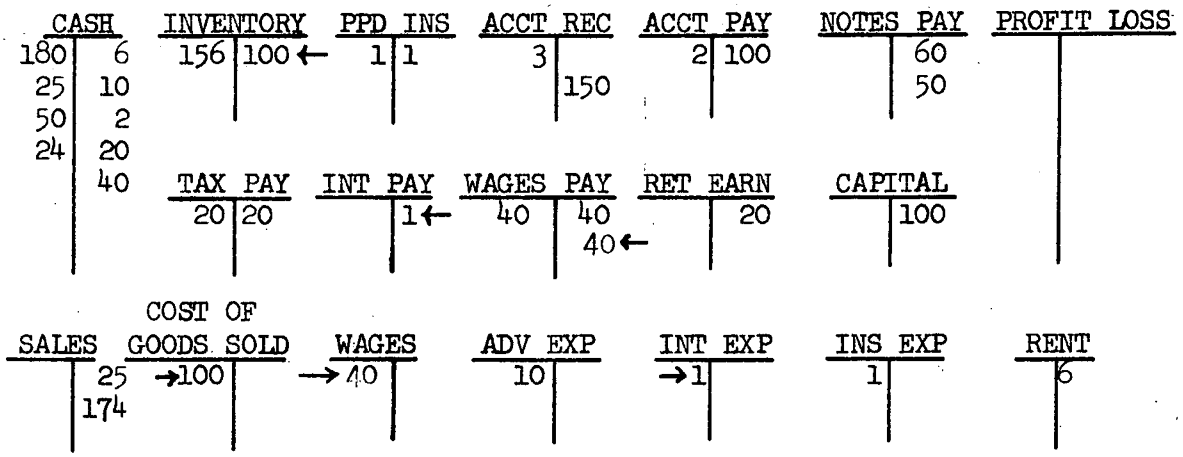
next example or question. CLOSE skips to EXAMPLE 1 <see (11)>. The screen is again full, and is erased when Kathy strikes new line.

CLOSE then presents the first question in the adjusting entries section <see (12)> in Fig. 14. Kathy misunderstands the question and gets the answer completely wrong. Kathy decides to seek help and types "help". CLOSE responds by printing out one of the accounts involved. With this prompting, Kathy types in an entry, but she still doesnt't know the rest of the answer. CLOSE responds "Your answer, dr cost of goods sold 100, is correct so far, type in the other entry." Kathy goes back to "help" and learns that the other account is inventory. With this bit of assistance, she gets the answer correct.

CLOSE then types another question <see (13)>, but Kathy decides to skip to the closing entries section, and types "skip" followed by a choice of option "b". Note that Kathy skipped only one question in this case; it involved INT PAY and INT EXP as can be seen from the entries posted to those accounts.

Kathy proceeds in this fashion until she finally finishes the closing entries section. At this point, CLOSE erases the screen, redraws and closes out the accounts, and presents a final balance sheet and income statement <see

Figure 14



(12) <u>QUESTION 1</u> : The firm sold $199,000 worth of Adzes during the month.
The stock sold cost $100,000.
READY dr cash 199 cr sales 199 (LF)
Kathy, your answer, dr **cash** 199 cr **sales** 199, has both entries incorrect.
If you need assistance, type "help"
READY help (LF)
One of the accounts is cost of goods sold, type in the entry.
READY dr cost of goods sold 100 (LF)
Your answer, dr cost of goods sold 100, is correct so far, type in the other entry.
READY help (LF)
The other account is inventory, type in the entry.
READY cr inventory 100 (LF)
Good, Kathy, that's correct.
<u>ANSWER</u> : dr cost of goods sold 100 cr inventory 100
READY (LF)

(13) <u>QUESTION 2</u> : The firm owes $40,000 to employees for wages earned during the month.
READY skip (LF)
(A) Print questions and answers so I know what I am skipping.
(B) Skip to the next section.
(C) Skip to the next question.

Type desired option : b (LF)

After all the adjusting entries have been made, a firm can <u>close</u> its books first, all <u>income</u> accounts are closed to the profit loss account
READY (LF)

Figure 15

CASH	
180	6
25	10
50	2
24	20
	40
	201
201	

INVENTORY	
156	100
	56
56	

PPD INS	
1	1
0	

ACCT REC	
3	153
150	
153	

ACCT PAY	
2	100
98	
	98

NOTES PAY	
110	60
	50
	110

PROFIT LOSS	
40	199
10	
1	
100	
1	
6	
41	

TAX PAY	
20	20
0	

INT PAY	
1	1
	1

WAGES PAY	
40	40
40	40
	40

RET EARN	
61	20
	41

CAPITAL	
100	100
	100

SALES	
119	25
	174

COST OF GOODS SOLD	
100	100

WAGES	
40	40

ADV EXP	
10	10

INT EXP	
1	1

INS EXP	
1	1

RENT	
6	6

BALANCE SHEET

ASSETS		
cash	201	
inventory	56	
ppd ins	0	
acct rec	153	
TOTAL ASSETS		410
EQUITIES		
LIABILITIES		
acct pay	98	
notes pay	110	
tax pay	0	
int pay	1	
wages pay	40	
TOTAL LIABILITIES		249
capital	100	
ret earn	61	
TOTAL EQUITIES		161
		410

INCOME STATEMENT

sales	199	
cost of goods sold	100	
GROSS PROFIT		99
OPERATING EXPENSES		
wages	40	
adv exp	10	
int exp	1	
ins exp	1	
rent	6	
TOTAL OPERATING EXPENSES		58
NET PROFIT		41

Fig. 15>.

When Kathy understands how these reports were prepared, she types new line and CLOSE responds "Thank you for using CLOSE. Good-bye ...." and returns control to CTSS.

CHAPTER VI

SUMMARY AND RECOMMENDATIONS

As stated in Chapter I, the ALP Project set for itself the following goals: 1) to find ways to make the learning process more efficient, 2) to tailor the learning process to the student's background and interests, 3) to integrate material across functional lines, and 4) to provide an experimental setting for research in the learning process. CLOSE concentrates on the first and second of these goals.

In particular, CLOSE is designed as a tutor for the student who has already been exposed to a "textbook" introduction to basic accounting. This "textbook" introduction may be the traditional classroom lecture, a programmed instruction text, or another CAI system designed to assist a student in learning basic accounting concepts. CLOSE attempts to provide the student with reinforcement and drill in basic accounting techniques, often initially providing

the student with a second view of conceptual material. CLOSE functions as a highly interactive, highly diagnostic tutor which uses graphics as well as text to present drill and conceptual material to the student. In addition, CLOSE allows the student a large degree of control over the program, thereby allowing the student to use the program as a self-test of his knowledge of accounting; he may answer or skip questions as he sees fit. He also has available to him diagnostics and additional help for any material which he finds difficult.

In essence, CLOSE attempts to provide the same concept reinforcement that the classroom provides in a "second view of the material" and a homework-drill sequence in traditional educational systems. However, the use of CLOSE frees valuable class time for the teacher, enabling him to present new conceptual material, or to discuss more fully material already presented. In addition, CLOSE is concerned with helping the student to learn material more efficiently. In particular CLOSE:

1) allows the student to advance at his own rate. The bright student is not held back, nor is the below average student forced to learn at a rate above his ability.

2) permits the student to tailor the time he spends learning according to his needs; he may concentrate on the section(s) where he needs additional tutoring and skip the rest.

3) presents the subject material in a highly efficient, highly concentrated, individually-oriented tutorial session. There is a substantial amount of wasted time in a classroom when any particular student is not learning. CLOSE attempts to keep each student learning throughout the session.

4) removes the boring arithmetic manipulations and T-account-drawing-and-ruling normally required of the student. These can be much more efficiently handled by the computer.

5) gives the student control of his tutorial; he may answer or skip questions. Diagnostics and help are also available.

6) allows the student to learn at a time and place convenient for him.

7) provides the student with his own tutor.

While it is difficult to describe in a few words all that CLOSE does, an accurate summary is that CLOSE provides <u>student-centered</u> <u>reinforcement</u> <u>of</u> <u>techniques</u>. a time-sharing system for immediate response, and graphics help to

CLOSE has not yet been used by a class of graduate students. However, it has been used, as a teaching program, by a group of eight students in the Sloan Fellows program at M.I.T. These students used the program and filled out questionaires. Their responses, while far from being statistically significant, give an indication of the type of response CLOSE will meet when used in the classroom. The majority of the Sloan Fellows group found the program "useful" (mean 2.4, variance 1.0 on a 1-10 scale with 1 being very useful). They tended to prefer this method of instruction to the standard programmed instruction textbook (mean 2.0, variance 1.2). When asked if they had "learned anything from the program", the group, as a whole, felt that they had (mean 2.6, variance 2.3). They found the response CLOSE gave when their answer was incorrect, "useful" (mean 2.2, variance 1.4). However, they were divided on the usefulness of the "help" command (mean 2.8, variance 6.2) and the "skip" command (mean 2.3, variance 9.4). Several students stated that they had not used either the "help" or the "skip" command, and, consequently, gave them low ratings. They were also divided on who controlled the program, they or CLOSE (mean 4.4, variance 7.7). On the average, the group found the response time "about right" (mean 5.1, variance 2.4 where 1 is too fast and 10 too slow). CLOSE is currently an operational accounting tutor, but there are many ways CLOSE can be improved and expanded.

In particular, it would be useful to expand the "skip" and "help" commands. "Skip" could be modified to allow the student to jump backward as well as forward in the program. This would allow him to review the material covered. "help" could be changed to give conceptual "help" in answering a question. For example, "help" would explain or rephrase the question and not just print out part of the answer.

Another improvement would be to expand the scope of the program to include financial statement analysis. Concepts presented could include profitability tests (e.g. gross profit percentage, return on sales, etc.), liquidity tests (e.g. current ratio, receivables to sales, etc.), solvency tests (e.g. equity ratio, times interest earned, etc.), and general overall measures (e.g. return on investment, earnings per share, etc.). If such a section was added, the student would complete the program wit an even broader view of the accounting process.

Another modification would be to add a decision rule to CLOSE. If the student gets the first five ledger entries wrong, he needs practice; give him more than usual. On the other hand, if he gets the first five right, perhaps it would be best to let him go on to the adjusting entries. The implementation of such a rule does not appear to be extremely difficult. The main problem seems to be the determination of an effective decision rule, i.e. should the program key on total number, consecutive number, or percentage right or wrong in deciding what to do next?

In an effort to free the instructor from the programming problems of CLOSE, it would be useful to develop a program which, when called, makes changes in CLOSE, compiles and loads it automatically. Changes could be: add a question, delete a question, change a question, change the order of questions, or change initial balances in accounts. If such a program were developed, it would make the teacher independent of CLOSE's inner workings; he could call a program which would guide him in making changes.

APPENDIX A

SYSTEM DOCUMENTATION

Location of CLOSE:

CLOSE is stored on disc in CLOSE SAVED in M5835 224. From within M535 224 the system may be started by typing "R CLOSE 1" followed by new line (LF). From another file directory (on CTSS) type "LINK CLOSE SAVED M5835 224 (LF)"; then type "R CLOSE 1 (LF)".

CLOSE will create (or append to already existing files) the two system monitor files CLOSES STATUS and TUTORS STATUS.

CLOSE Files:

The following files are used in conjunction with the CLOSE system:

CLOSE SAVED	Core image of CLOSE
CLOSES STATUS	System monitor
TUTORS STATUS	System monitor
MASTER ALGOL	Source code
CLOSE LOAD	Load file for CLOSE routines

ARDS LOAD	Load file for ARDS routines
AIMSLB LOAD	Load file for AIMSLB routines
MYLIB SQZBSS	Simulated load file

<u>Loading CLOSE</u>:

Starting from a source deck, CLCSE is converted to a running program in the following manner:

1) Compile all routines in MASTER ALGOL. This is accomplished by typing "TAED PRCGRAM.NAME * (LF)" for each PROGRAM.NAME in MASTER ALGOL. Successful compilation creates a PROGRAM.NAME BSS file.

2) Compress all BSS files. Type "SQZBSS PROGRAM.NAME (LF)" for each PROGRAM.NAME.

3) Create a LODSIM file. Type "DO P LODSIM MYLIB CLOSE (GET) CLOSE (GET) AIMSLB (GET) ARDS (LF)". Successful execution of this command creates a file, MYLIB BSS, which contains the entry points for each system procedure used in CLOSE. Compress MYLIB BSS by typing "SQZBSS MYLIB (LF)". This creates MYLIB SQZBSS.

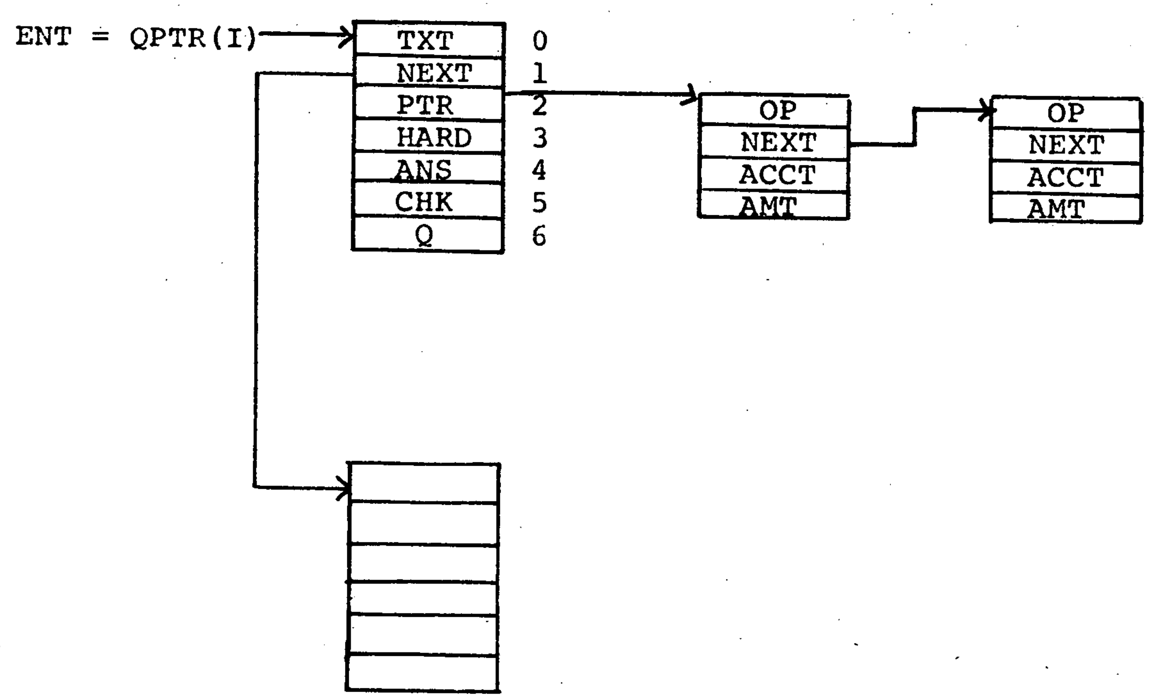
4) Load CLOSE by typing "LAED VLOAD CLOSE (GET) CLOSE (SQZ) AIMSL1 MYLIB (LF)"

Bead Structure

The bead structure concept plays an important role in the CLOSE system. The question bead and the account bead are the most important beads in the program. They are described in detail in this section. In the description of these beads they are shown in unpacked form, i.e. each component occupies a full word of storage, to facilitate an understanding of the structure. In actuality, all components of beads occupy fractional parts of a full word. This so-called packed form is also shown. All the other beads used in the program are relatively simple and are shown only in the packed form.

Figure 16

QUESTION AND ANSWER BEADS STRUCTURE



The components are defined as follows:

TXT Is a pointer to the text of the current question.

NEXT Is a pointer to the next question.

PTR Is a pointer to the first answer bead (see Answer Bead Structure).

HARD Is an integer which, if non-zero, indicates that the question is difficult. Diagnostics and error messages take this fact into account.

ANS Is a pointer to the text of the answer.

CHK Is an integer indicating how the answer is to be checked. Possible values are:

- 0 no check made
- 1 check answer
- 2 form answer, do not check (as in a closing entry when the program calculates the answer)

3 form answer and check

Q Is an integer giving the current question number. The numbering scheme is as follows:

1) the first digit, 1, 2, or 3, is the current phase of the program (1 stands for ledger entry, 2 adjusting entry, and 3 closing entry).

2) the remaining four digits are the relative question/example numbers within that phase, i.e. Q20030 is the third question/example in the adjusting entry stage.

Answer Bead Structure

OP Is the type of operation being performed, i.e. debit or credit.

NEXT Is a pointer to the next answer bead.

ACCT Is the account to be debited or credited.

AMT Is the amount to be debited or credited.

Note:

The concept of a bead is basic to the understanding of the program. TXT,NEXT,PTR, HARD,ANS,CHK,Q,OP,ACCT, and AMT are all components of a bead. To refer to any part of a bead, the component and the pointer must be given; the concept is similar to that of relative addressing.

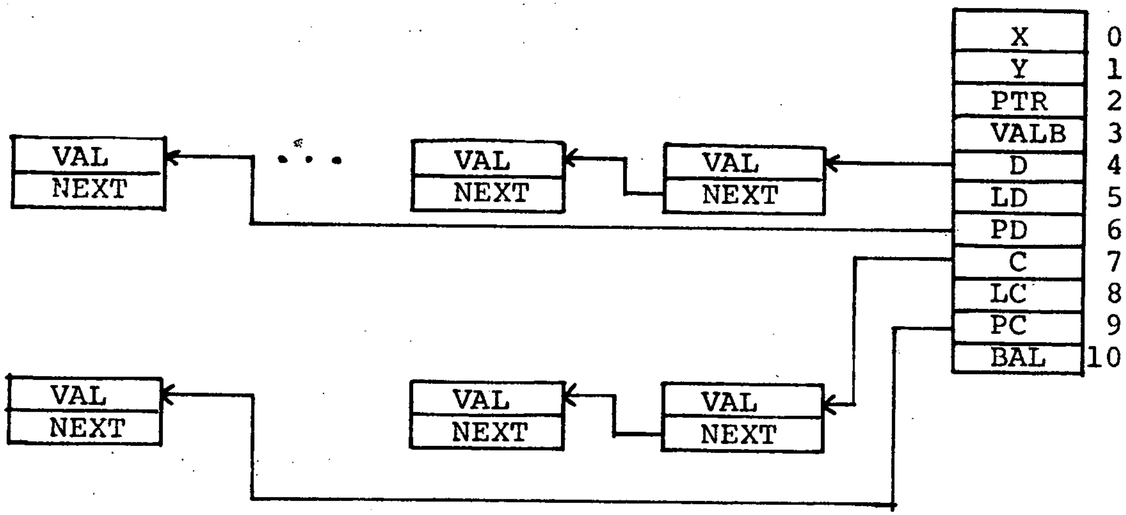
Example:

TXT(ENT) refers to the text of the question pointed to by ENT.

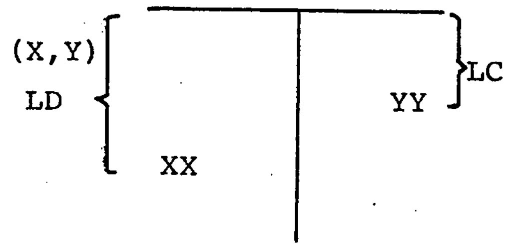
Q(ENT) refers to the question number of the question pointed to by ENT.

Figure 17

ACCOUNT BEAD STRUCTURE



a T-Account



The components of A are defined as follows:

X,Y	Are the X,Y coordinates of the account specified by A(I).
PTR	Is a pointer to its image in the display file (necessary when the light pen is used, not implemented in version SSM3).
VALB	Is a .C. pointer to the spelling of the account name.
D	Is a pointer to the <u>first</u> debit entry.
LD	is the Y distance to be stepped off before printing the current debit entry.
PD	Is a pointer to the storage allocated for the next debit entry. This could be determined from D and NEXT, but the use of PD is much more efficient.
C,LC,PC	Are, respectively, the same as D,LD, and PD, except that they are used for credit entries.

BAL Is the current balance of the account, initially zero as FREZ not only frees, but also zeros storage.

VAL Is the amcunt to be posted.

NEXT Is the pointer to the <u>next</u> two free words in storage.

Array A:

Array Component	Account
A(1)	CASH
A(2)	INVENTORY
A(3)	PPD INS
A(4)	ACCT REC
A(5)	ACCT PAY
A(6)	NOTES PAY
A(7)	PROFIT LOSS
A(8)	TAX PAY
A(9)	INT PAY
A(10)	WAGES PAY
A(11)	RET EARN
A(12)	CAPITAL
A(13)	WAGES
A(14)	ADV EXP
A(15)	INT EXP
A(16)	COST OF GOOD SOLD
A(17)	INS EXP
A(18)	RENT
A(19)	SALES

Figure 18

PACKED BEAD STRUCTURE

ACCOUNT BEAD

X				0
Y				1
	PTR		VALB	2
	D		LD	3
	PD		C	4
	LC		PC	5
BAL				6
S-1-2	3	18	21 35	

S-1-23 18 21 35

QUESTION BEAD

	TXT		NEXT	0
	ANS		Q	1
	PTR	CHK	HARD	2

POST BEAD

VAL		NEXT	0

Figure 19

PACKED BEAD STRUCTURE

ANSWER BEAD

		OP	NEXT	0
	ACCT		AMT	1

RT.PTR BEAD AND ACT.PTR BEAD

	WHOLE			0

ERROR BEAD

	WHOLE		NUM	0

D.ER	C.ER	ACT.ER	AMT.ER	FMT.ER	0

Documentation:

The following section describes each routine used in CLOSE. The location, calling sequence, function, called procedures, and flow charts are given for each procedure.

Procedure: AC.DMP

Location: AC.DMP ALGOL

Calling Sequence: AC.DMP(ACT.PTR,ENT)

where

ACT.PTR Is a pointer to a list of answer beads which form the answer to the current question.

ENT Is a pointer to the current question.

Function:

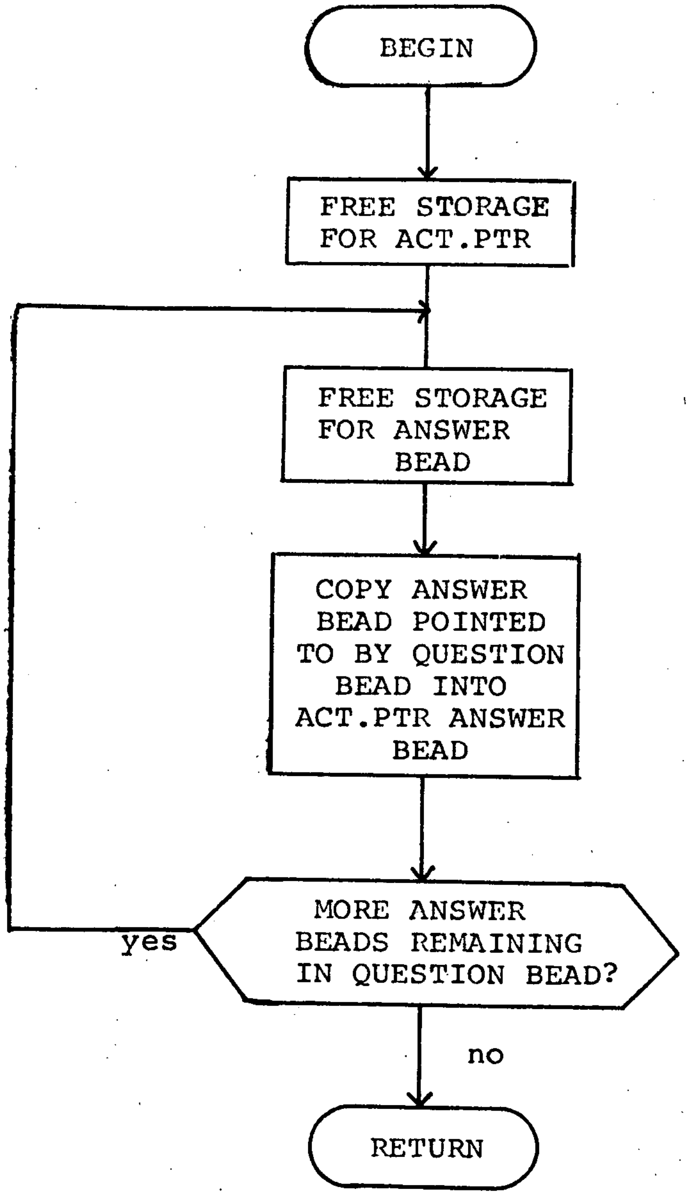
AC.DMP makes a copy of the list of answer beads for the current question. ACT.PTR is the pointer to the new list, and the length of the list is recorded as a component of ACT.PTR. The list pointed to by ACT.PTR will be used to check the student's answer.

Called Procedures:

Name	Location
FREZ	AIMSLB ALGOL

Figure 20

Flow Chart for AC.DMP



Procedure: ACT.CK

Location: ACT.CK ALGOL

Calling Sequence:

ACT.CK(M,ACT.PTR,RT.PTR,ERROR,TYPE,ITEM,ENT,DR.CR)

where

M — Is an integer specifying whether the student debited or credited (0 for debit, 1 for credit) the account currently being checked. The value of M is determined by a call on D.C.CK.

ACT.PTR — Is a pointer to a list of answer beads which have not yet been given as an answer to the current question.

RT.PTR — Is a pointer to a list of answer beads which the student has answered correctly.

ERROR — Is a pointer to the error bead. If any errors are discovered, the types and number of errors are recorded in ERROR.

TYPE	Is an integer indicating the TYPE of the ITEM, i.e. integer, alphabetic, etc. (see procedure RD), read in from the console.
ITEM	Is a pointer to the ITEM read in from the console.
ENT	is a pointer to the current question.
DR.CR	Is a pointer to "dr" or "cr" established by D.C.CK.

<u>Function</u>:

ACT.CK is the main routine called by CHECK to determine if the student's answer is correct. ACT.CK looks through the accounts specified in ACT.PTR. It also checks the acccunts specified in RT.PTR as the part of the answer being checked may have been moved to RT.PTR by a previous partially correct answer. It attempts to match the account name with an account name in either ACT.PTR or RT.PTR. An exact match is not required as ACT.CK uses a mispelling algorithm to cancel the effect of most common spelling errors.

If ACT.CK finds the account name in either ACT.PTR or RT.PTR, it checks to see if the account is being correctly debited or credited. It also calls AMT.CK to check the amount. If the three item group, i.e. dr or cr, account, and amount, is found to be correct, it is moved from ACT.PTR to RT.PTR.

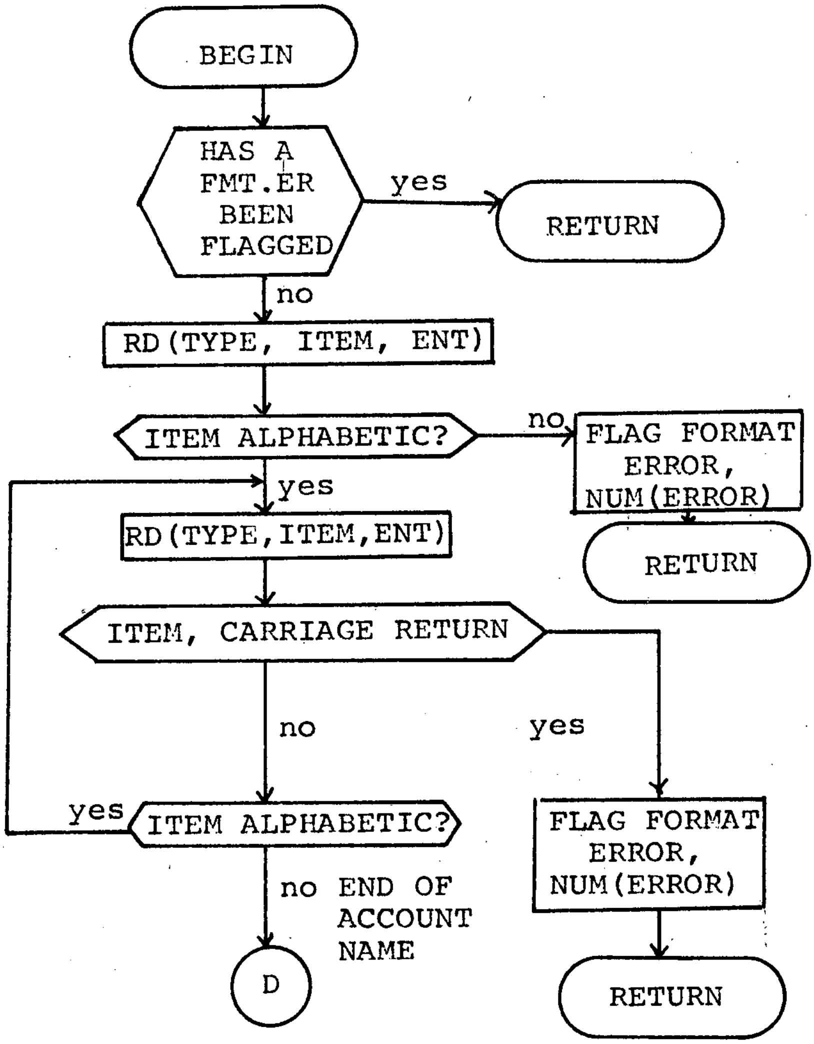
If a group is found to have an error, i.e. wrong account, incorrect debit or credit, wrong amount, the incorrect item(s) of the group is surrounded by asterisks and flags are set indicating the types and number of errors found.

Called Procedures:

Name	Location
AEQL	AEQL ALGOL
AMT.CK	AMT.CK ALGOL
ASC.C.	AIMSLB ALGOL
ASCINT	" "
ASCOUT	" "
ASCSAV	" "
CFRET	" "
CPY.LN	CPY.LN ALGOL
CVTOIN	AIMSLB ALGOL
MISPEL	MISPEL ALGOL
NEWLIN	AIMSLB ALGOL
RD	SETBUF ALGOL
WFMT	WFMT ALGOL

Figure 21

Flow Chart for ACT.CK



(See page 101)

Figure 21 continued

FROM PAGE 100

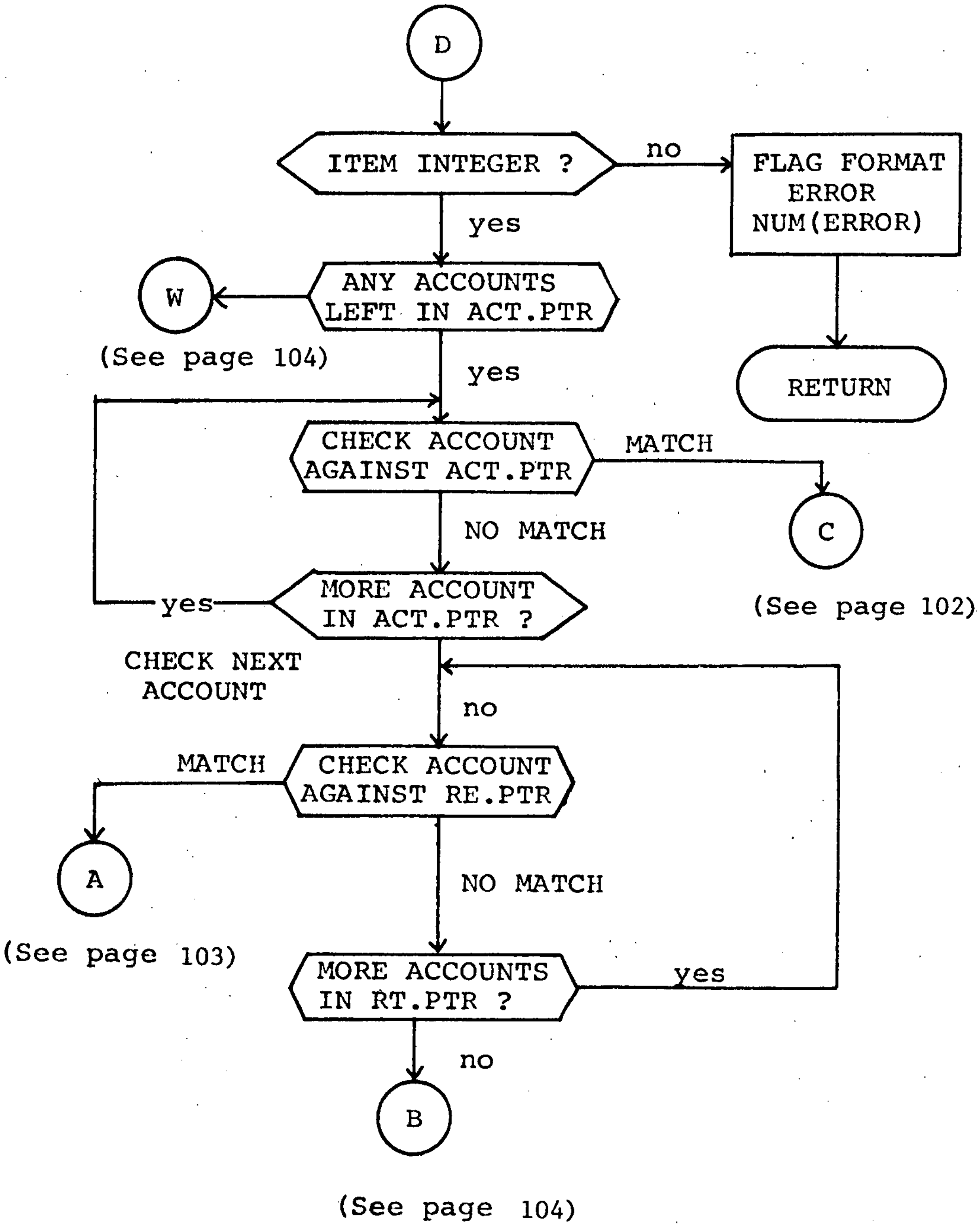


Figure 21 continued

FROM PAGE 101

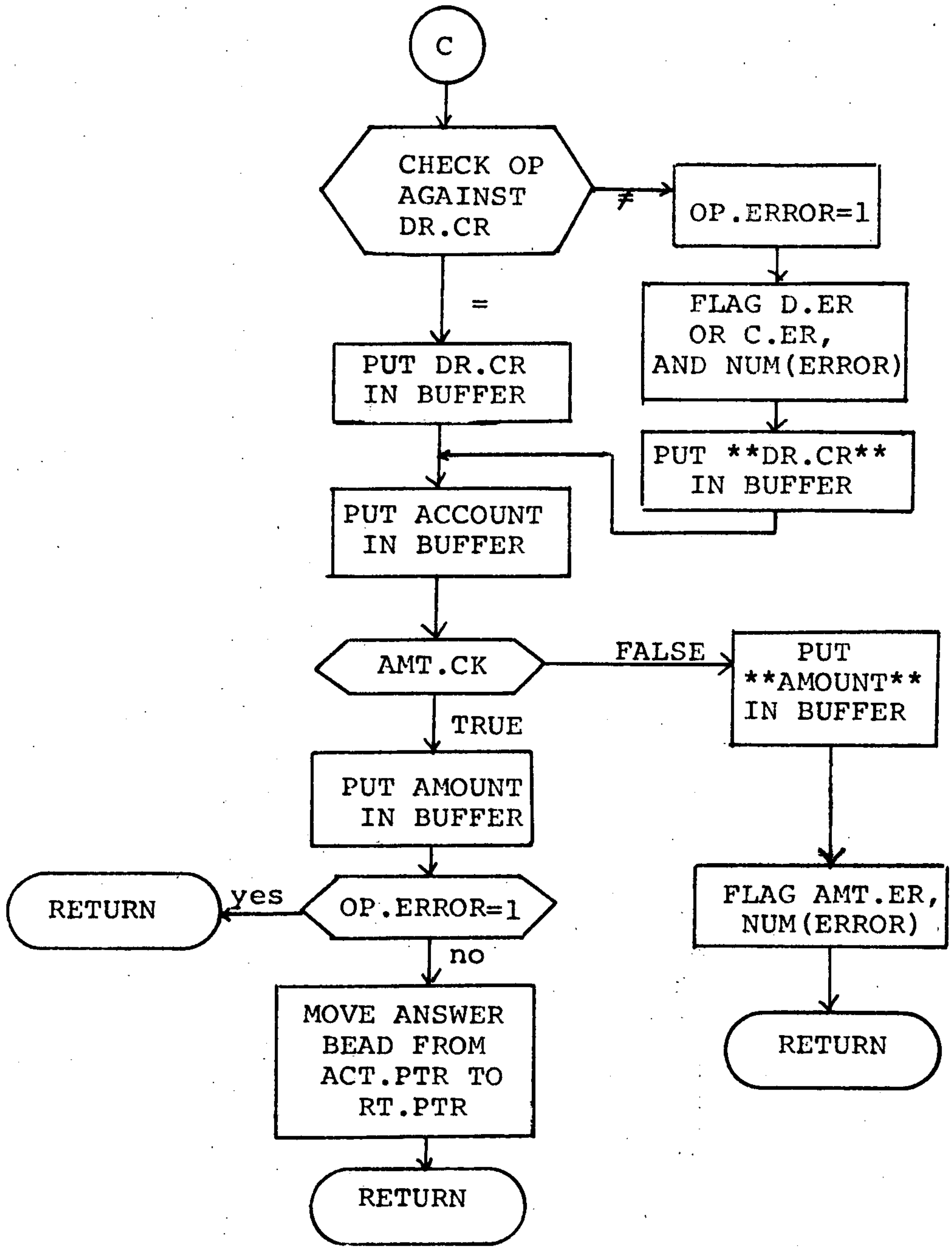


Figure 21 continued

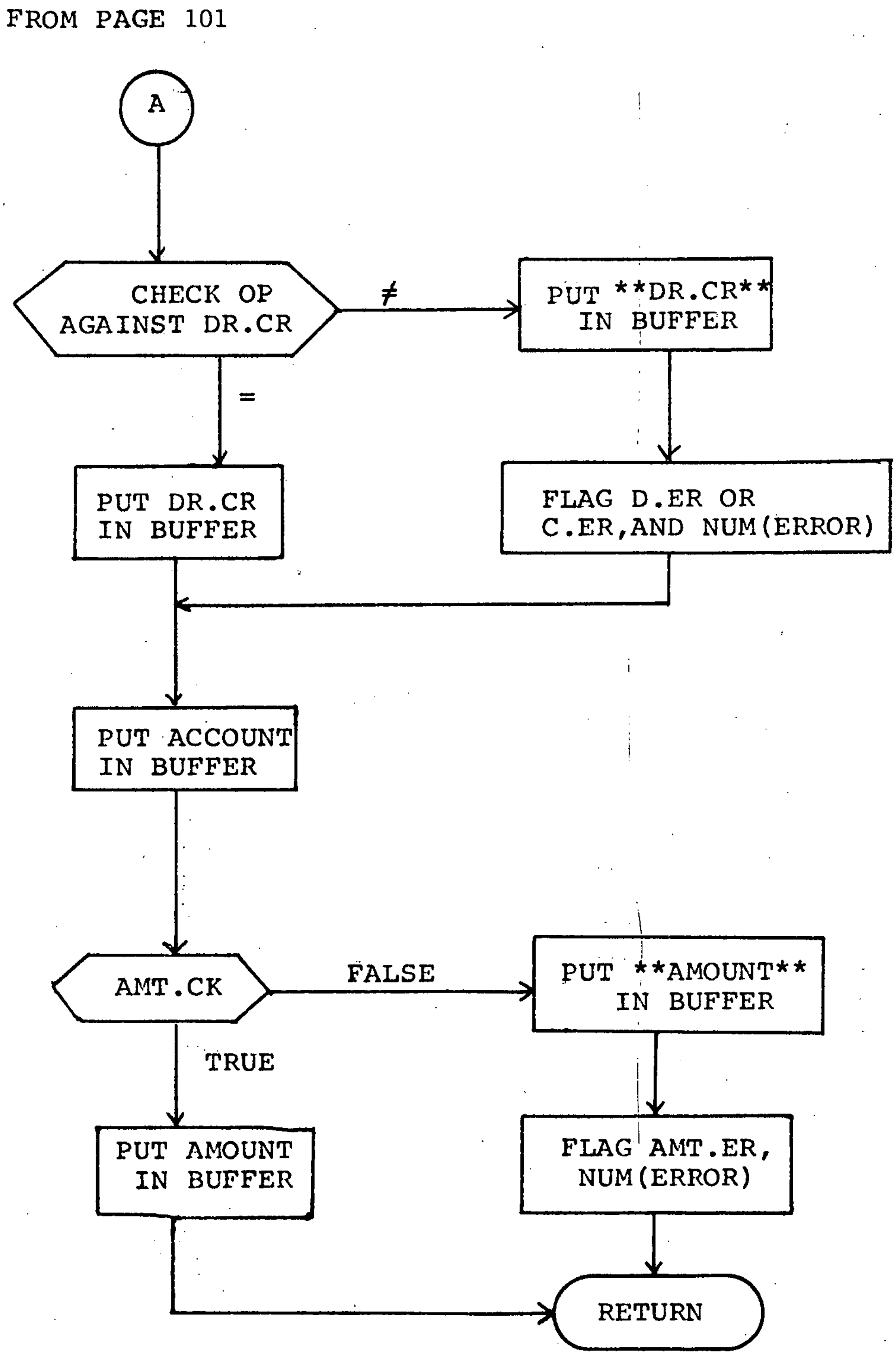
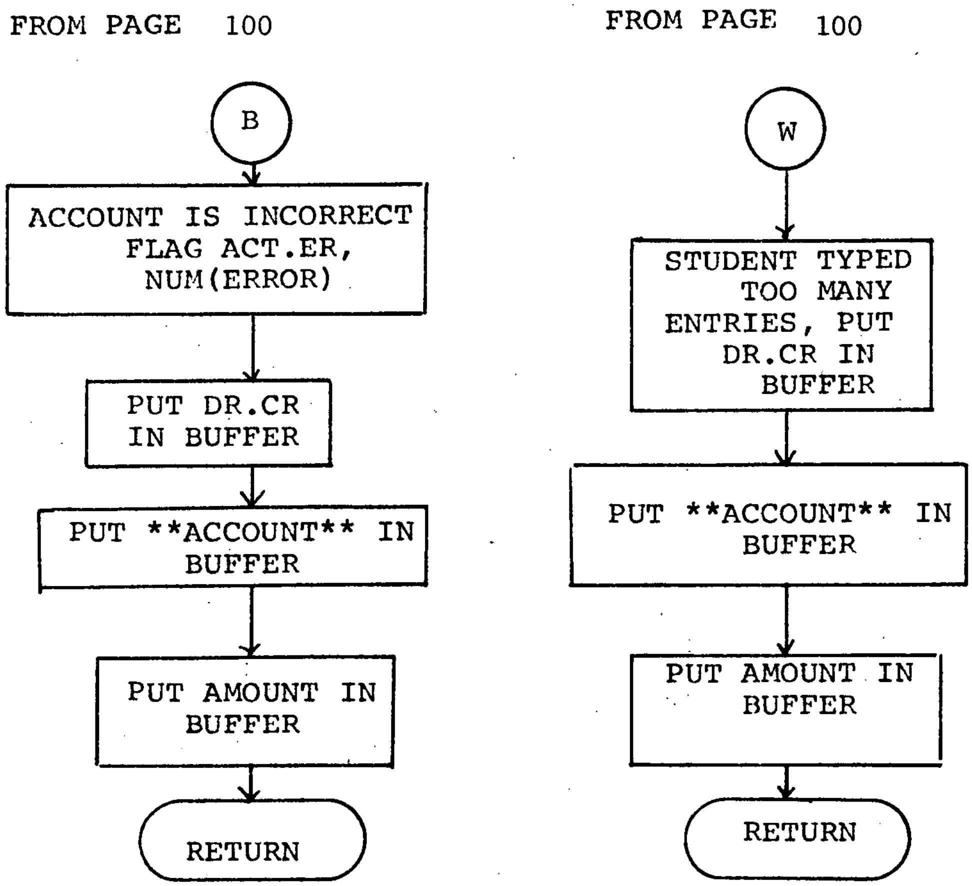


Figure 21 continued



Procedure: ADJ.1

Location: ADJ.1 ALGOL

Calling Sequence: ADJ.1(A,QPTR)

where

A Is an array whose elements are pointers to the various T-accounts.

QPTR Is an array whose elements are pointers to the various questions.

Function:

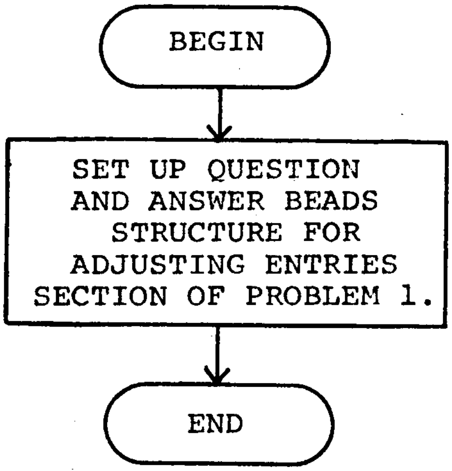
This procedure sets up the question and answer beads for the adjusting entries section of problem 1.

Called Procedures:

Name	Location
FREZ	AIMSLB ALGOL

Figure 22

Flow Chart for ADJ.1



Procedure: AEQL

Location: AEQL ALGOL

Calling Sequence: AEQL(STRNG1,STRNG2)

where

STRNG1, STRNG2 are pointers to the strings to be compared.

Function:

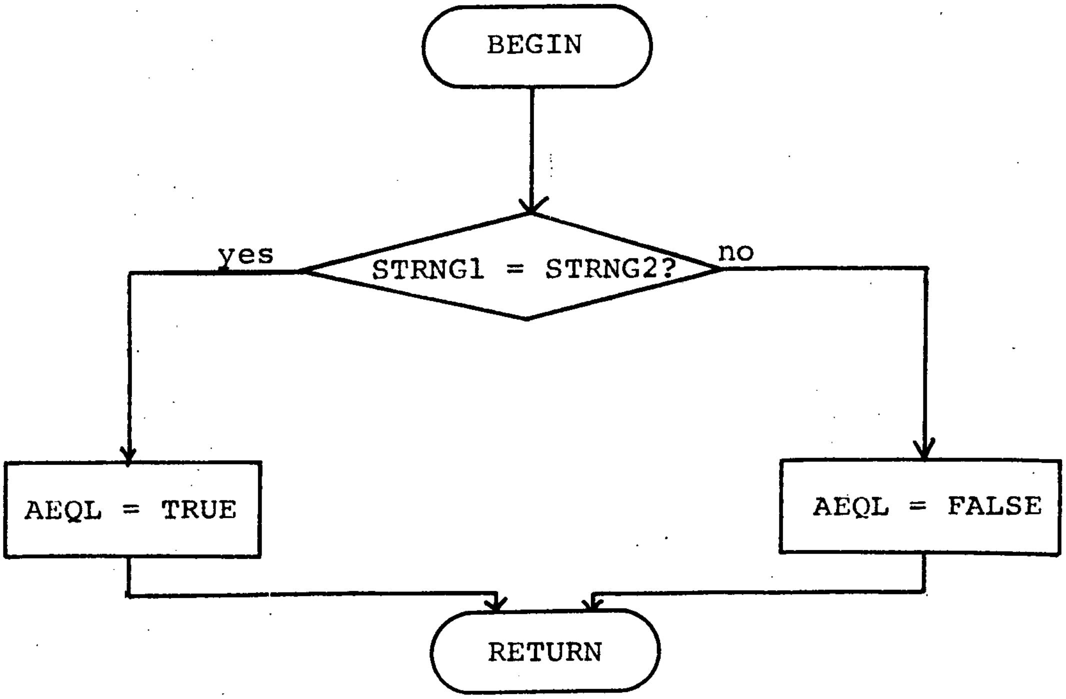
AEQL compares two strings according to the usual alphabetic conventions; no distinction is made between upper and lower case. If the strings are equal, the value of AEQL is TRUE; otherwise, it is FALSE.

Called Procedures:

Name	Location
ACOMP	AIMSLB ALGOL

Figure 23

Flow Chart for AEQL



Procedure: AMT.CK

Location: AMT.CK ALGOL

Calling Sequence: AMT.CK(ENT,ITEM)

where

ENT Is a pointer to the current answer bead.

ITEM Is a pointer to the current item, i.e. the item most recently retrieved from the student's input buffer.

Function:

The procedure AMT.CK checks the amount typed in by the student against the correct amount for this part of the entry. If they agree, AMT.CK returns TRUE; otherwise, it returns FALSE.

AMT.CK disregards thousands (000s), i.e. 10, 10,000, and 10000 are all acceptable and are considered identical. All answers in CLOSE are in thousands of dollars. Consequently, the student may type in only the significant digits, or he may type the amount in

β

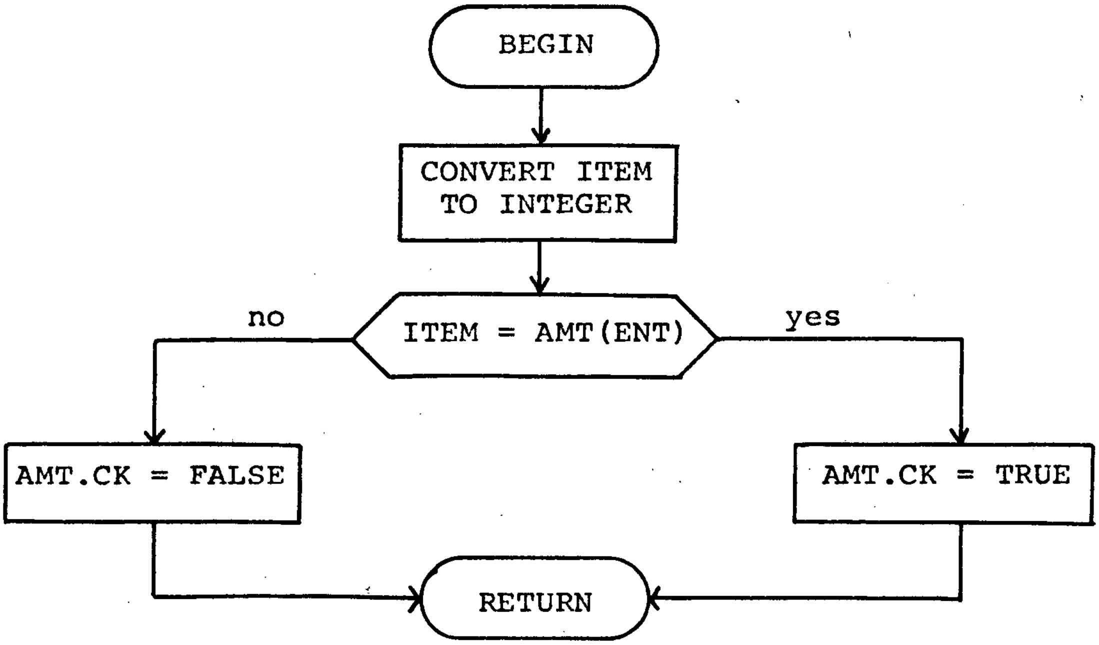
full.

<u>Called Procedures</u>:

Name	Location
CVTOIN	AIMSLB ALGOL

Figure 24

Flow Chart for AMT.CK



Procedure: CHECK

Location: CHECK ALGOL

Calling Sequence: CHECK(ENT,A,NAME1,ACT.PTR,RT.PTR)

where

ENT Is a pointer to the current question.

A Is an array whose elements are pointers to the various T-accounts.

NAME1 Is a pointer to the student's first name.

ACT.PTR Is a pointer to a list of answer beads which have not yet been given as an answer to the current question.

RT.PTR Is a pointer to a list of answer beads which the student has answered correctly.

Function:

This procedure checks the answer typed in at the console by the student. If the answer is correct, CHECK returns a value of 0; otherwise, it returns a value of 1. It accomplishes this by searching for a match between the accounts the student types in and the accounts of the answer beads. If CHECK can match accounts, it then checks for a match with OP and AMT (see answer bead on p. xxx). CHECK calls procedures D.C.CK, ACT.CK, and AMT.CK to do the actual checking of each individual item.

In addition, CHECK provides for the dynamic determination of closing entry answer beads. CHECK forms each answer bead so that every income and expense account can be properly closed. It also forms the correct answer to be output to the student.

Called Procedures:

Name	Location
ACT.CK	ACT.CK ALGOL
AEQL	AEQL ALGOL
ASC.C.	AIMSLB ALGOL
ASCINT	" "
ASCOUT	" "
ASCSAV	" "
ASCTAB	" "
ASCWRP	" "
CFRET	" "

D.C.CK	D.C.CK ALGOL
FRET	AIMSLB ALGOL
FREZ	" "
HELP	HELP ALGOL
NEWLIN	AIMSLB ALGOL
RD	SETBUF ALGOL

Figure 25

Flow Chart for CHECK

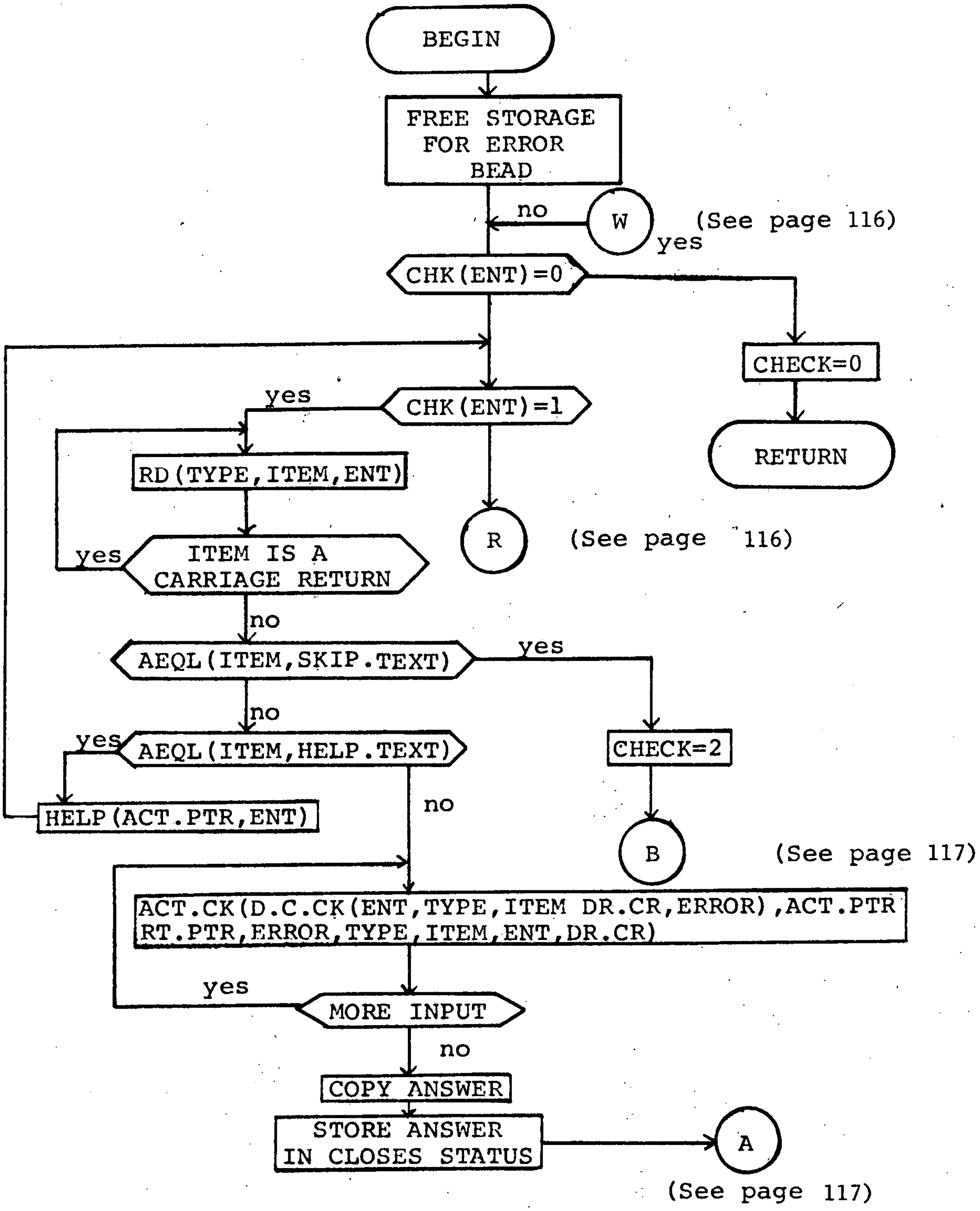
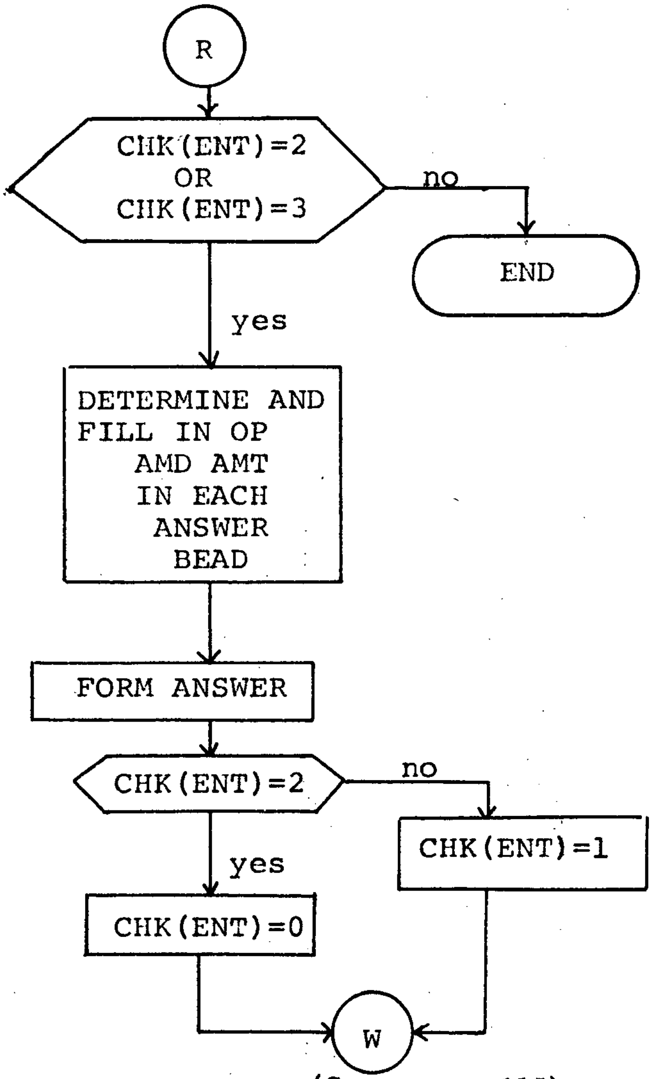


Figure 25 continued

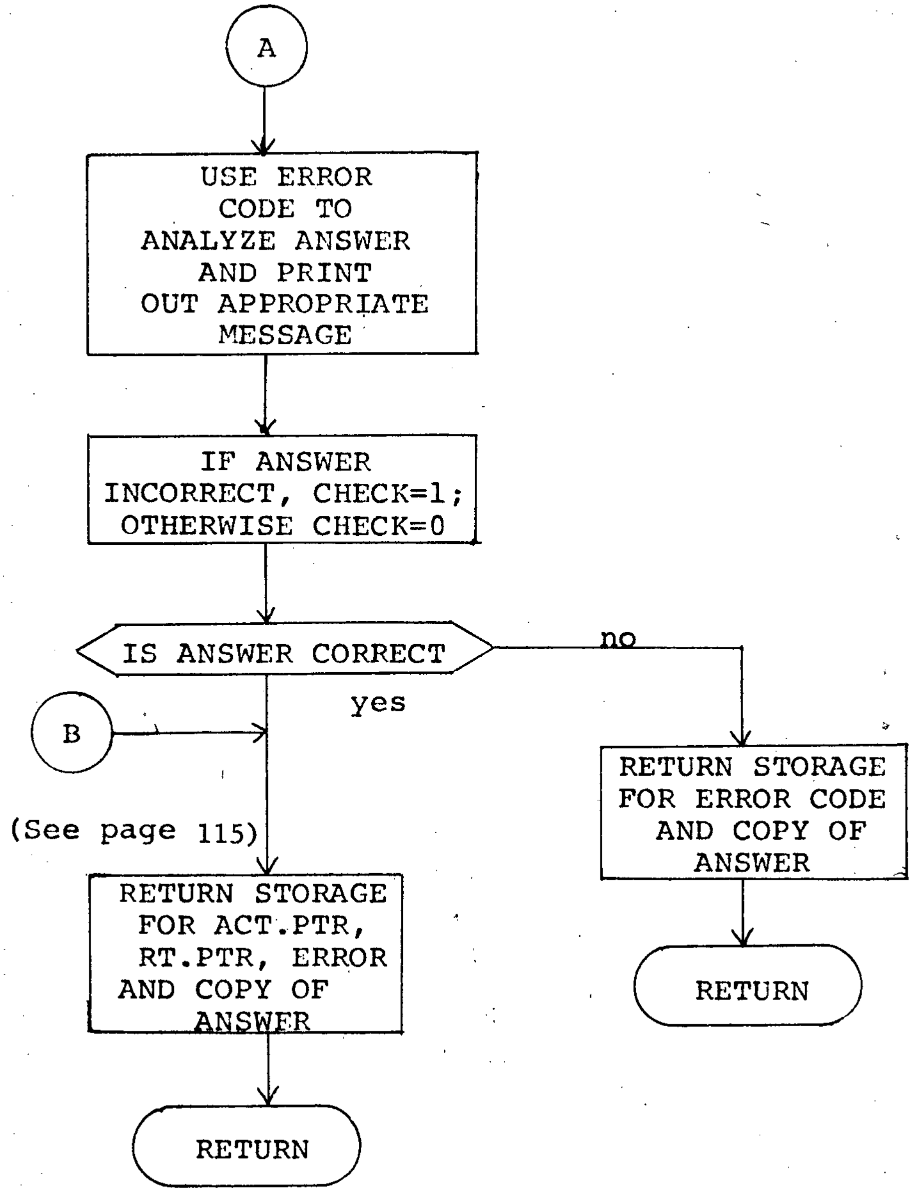
FROM PAGE 115



(See page 115)

Figure 25 continued

FROM PAGE 115



Procedure: CHOOSE

Location: CHOOSE ALGOL

Calling Sequence: CHOOSE(K)

where

K Is a pointer to a list of available options.

Function:

CHOOSE allows the student to be given a choice on the flow of the program; the student is presented with the available options, pointed to by K, and after he types in the letter corresponding to his choice, CHOOSE returns the number of the option typed, i.e. the third option, or the first option, etc. The program can then proceed in the desired manner.

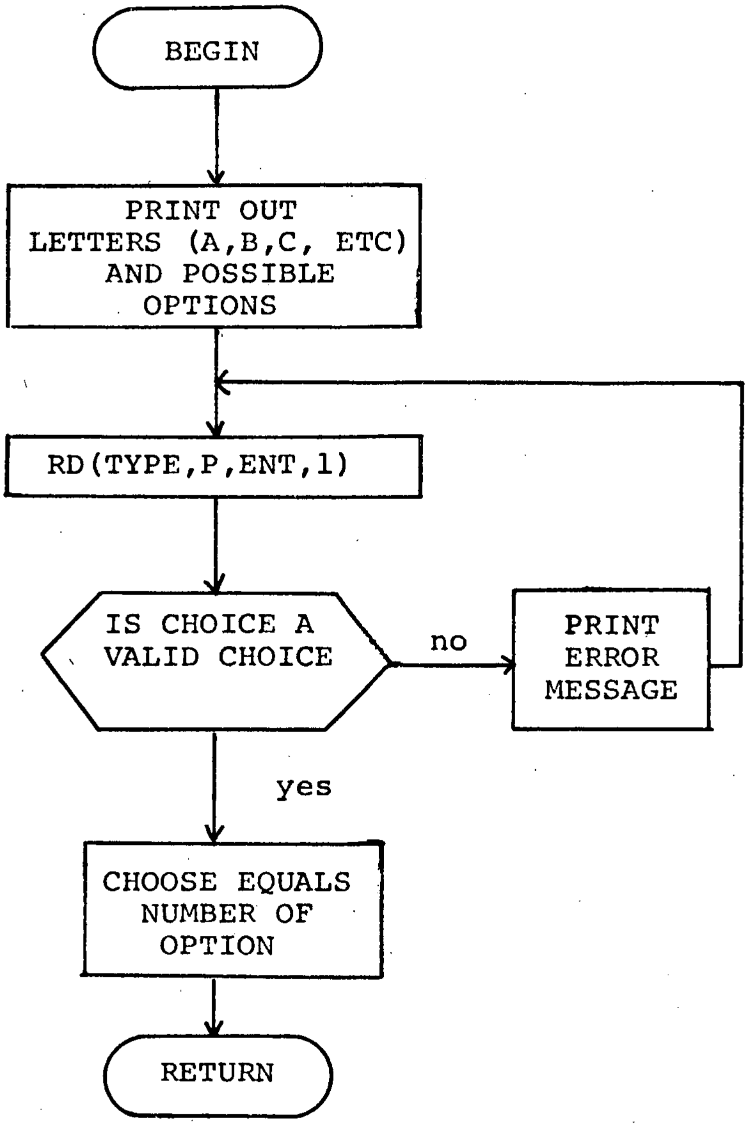
Called Procedures:

Name	Location
ASC.C.	AIMSLB ALGOL
ASCAUT	" "
ASCOUT	" "

NEWLIN	"	"
RD	SETBUF	ALGOL

Figure 26

Flow Chart for CHOOSE



Procedure: CLO.1

Location: CLO.1 ALGOL

Calling Sequence: CLO.1(A,QPTR)

where

A — Is an array whose elements are pointers to the various T-accounts.

QPTR — Is an array whose elements are pointers to the various questions.

Function:

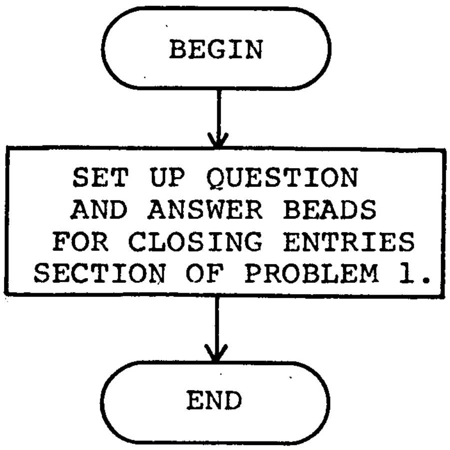
This procedure sets up the question and answer beads for the closing entries section of problem 1.

Called Procedures:

Name	Location
FREZ	AIMSLB ALGOL

Figure 27

Flow Chart for CLO.1



<u>Procedure</u>: CLOSE

<u>Location</u>: CLOSE ALGOL

<u>Calling Sequence</u>: -----------

<u>Function</u>:

This is the main program. It supervises the flow of control between all procedures. Because of the modular use of procedures and the use of bead structures, the main program is quite simple.

There are two main tasks which it has to handle: 1) initialization, and 2) program control (question-answer).

First, CLOSE initializes the account beads and the question beads. Next, CLOSE calls a sequence of procedures which print a heading, an introduction, finish initialization, set screen parameters, draw the T-accounts on the screen, and print initial financial reports on the screen. Now, CLCSE is ready for questions. CLOSE prints out a question (or example or text), then calls CHECK to evaluate the student's response, if any. Depending on CHECK's evaluation,

c

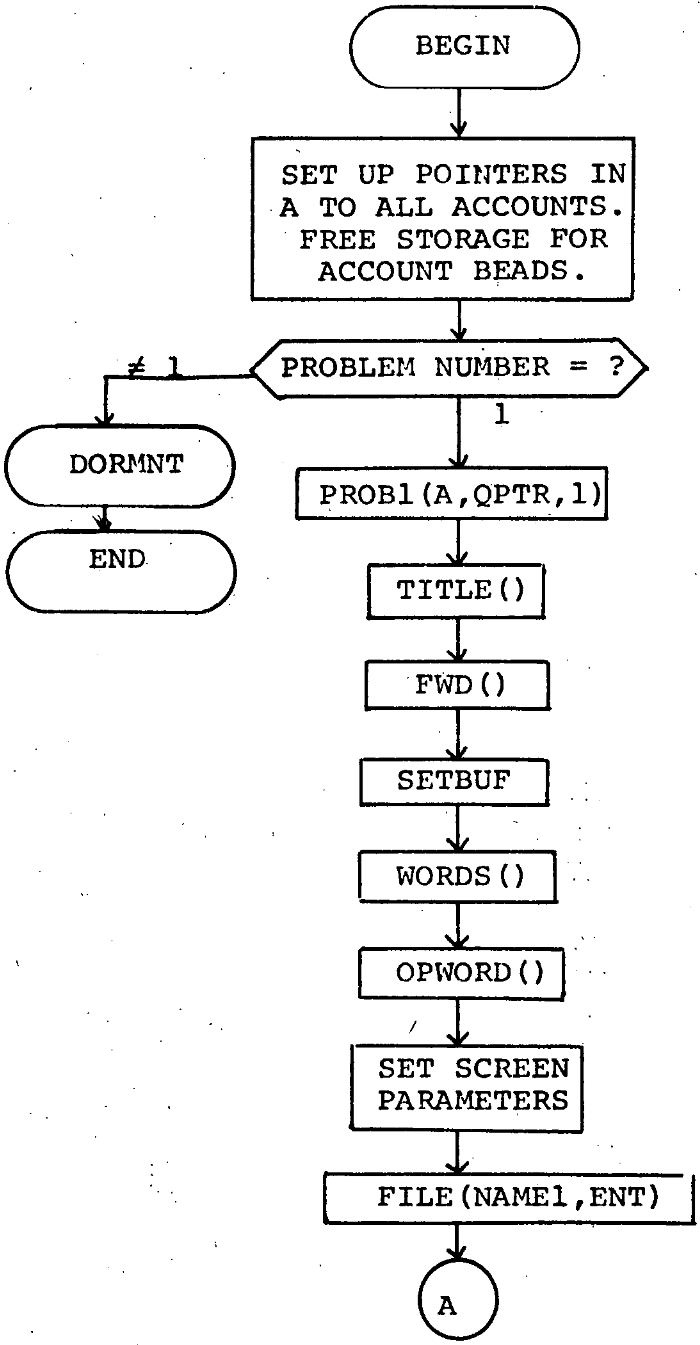
CLOSE either asks the student to try again as his answer is incorrect, or it posts the transaction through a call to MLTPST, prints out the answer, and goes on to the next question. CLOSE procedes in this fashion until it runs out of questions, indicated by NEXT(ENT)=0. Then it prints out the final reports and stops.

Called Procdures:

Name	Location
AC.DMP	AC.DMP ALGOL
ASC.C.	AIMSLB ALGOL
ASCOUT	" "
ASCWRF	" "
CHECK	CHECK ALGOL
DORMNT	CTSS
FILE	FILE ALGOL
FREZ	AIMSLB ALGOL
FWD	FWD ALGOL
LINCNT	LINCNT ALGOL
MLTPST	MLTPST ALGOL
NEWLIN	AIMSLB ALGOL
OPWORD	D.C.CK ALGOL
PROB1	PROB1 ALGOL
RD	SETBUF ALGOL
REPORT	REPORT ALGOL
RTMARG	AIMSLB ALGOL
SETPRM	ARDS
SGNON	"
SKIP	SKIP ALGOL
STOP	STOP ALGOL
TITLE	TITLE ALGOL
WORDS	CHECK ALGOL

Figure 28

Flow Chart for (MAIN)



(See page 126)

Figure 28 continued

FROM PAGE 125

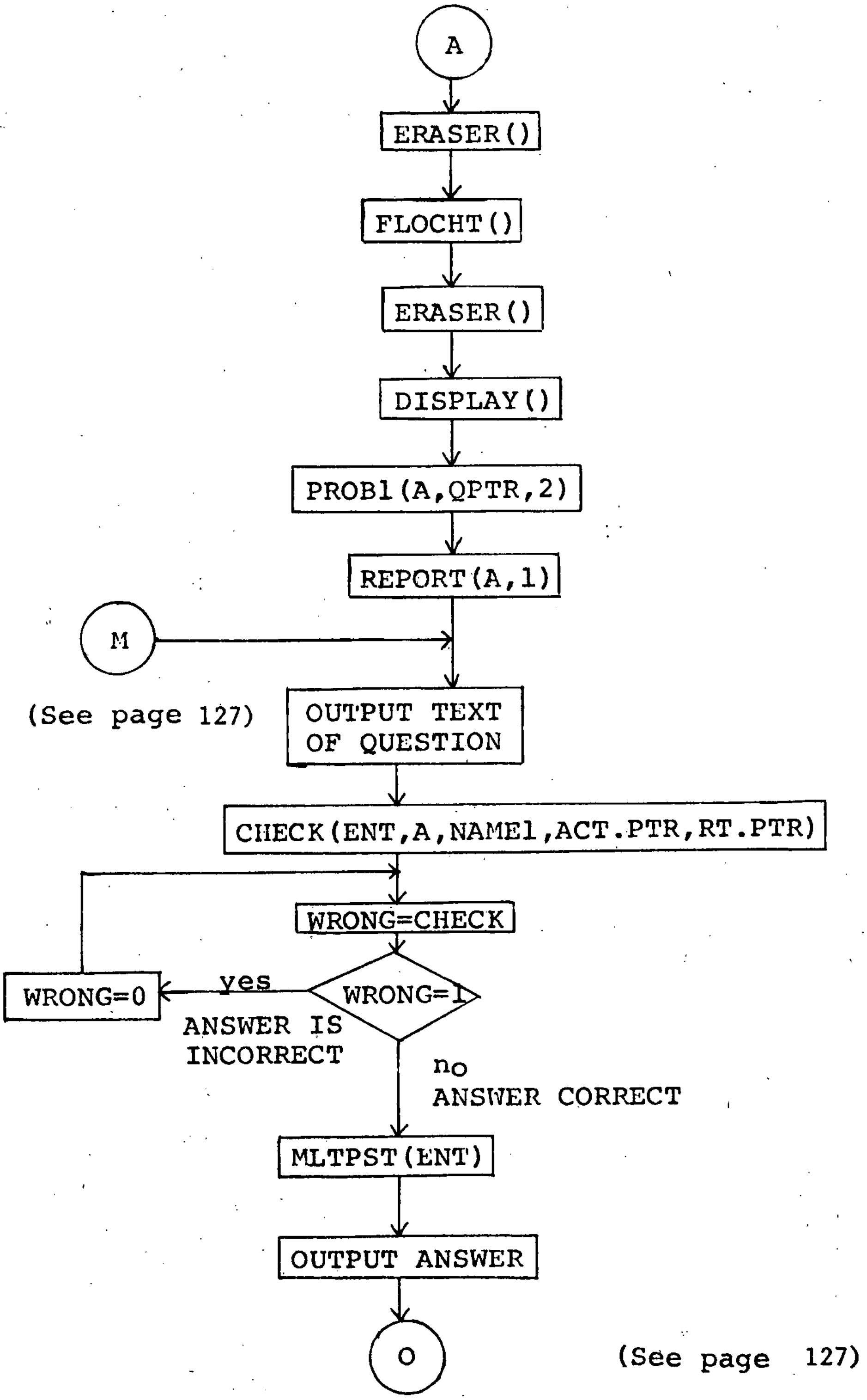
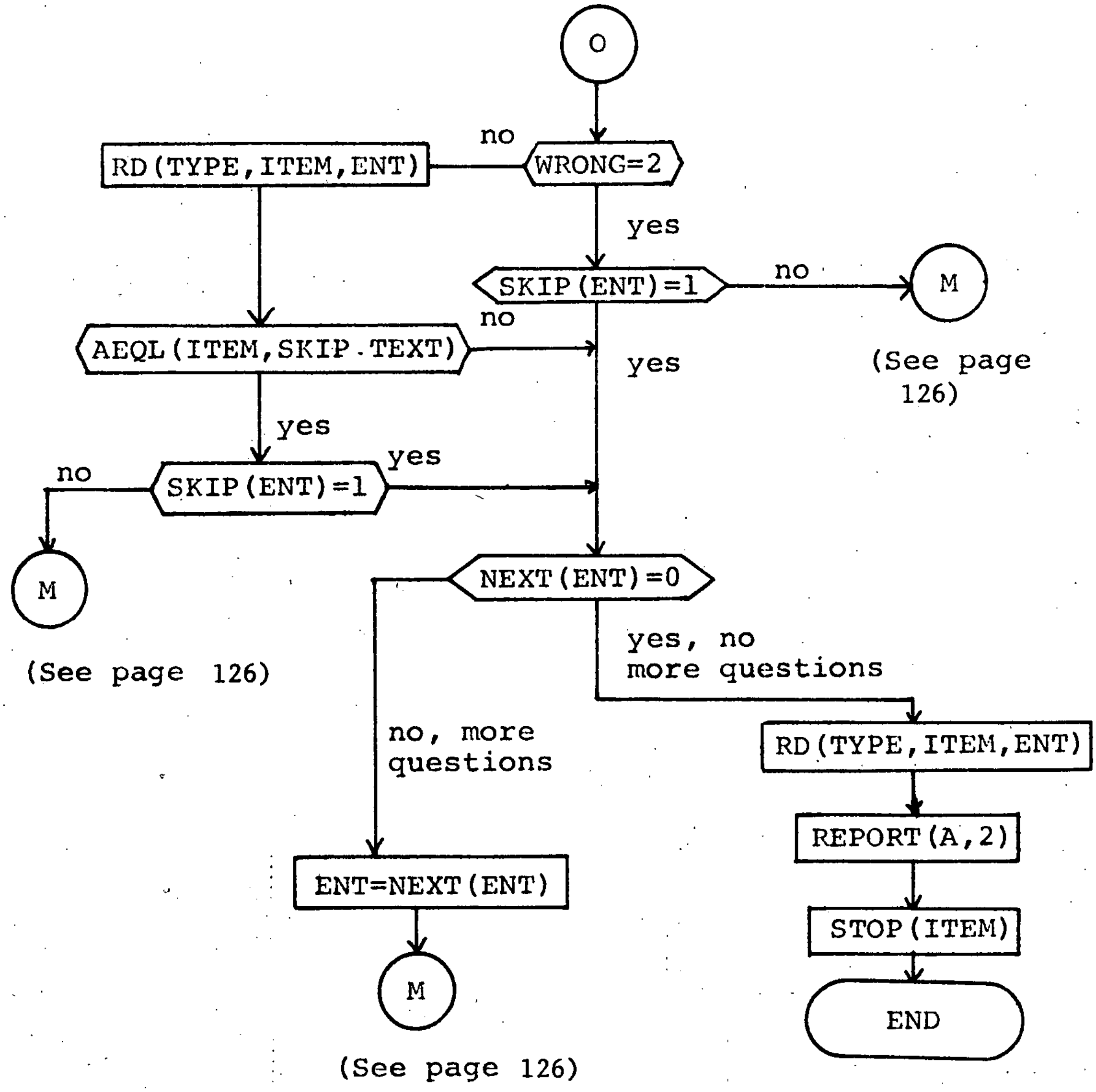


Figure 28 continued

FROM PAGE 126



Procedure: CPY.LN

Location: CPY.LN ALGOL

Calling Sequence: CPY.LN(ITEM)

where

ITEM Is a .C. pointer to the input line returned by CPY.LN.

Function:

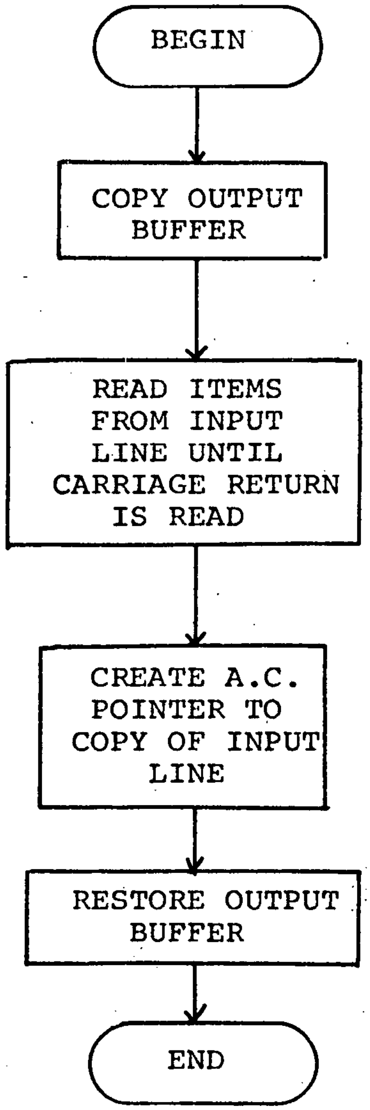
CPY.LN copies the input line from the console and returns a .C. pointer to the copy of the input line.

Called Procedures:

Name	Location
ASC.C.	AIMSLB ALGOL
ASCINT	" "
ASCSAV	" "
CVTOIN	" "
RD	SETBUF ALGOL

Figure 29

Flow Chart for CPY.LN



Procedure: D.C.CK

Location: D.C.CK ALGOL

Calling Sequence: D.C.CK(ENT,TYPE,ITEM,DR.CR,ERROR)

where

ENT Is a pointer to the current question.

TYPE Is an integer indicating the type of the ITEM, i.e. integer, alphabetic, etc. (see procedure RD), read in from the console.

ITEM Is a pointer to the item read in from the console.

DR.CR Is a pointer to the item if a debit or credit was found.

ERROR Is a pointer to the error bead. If any errors are discovered, the type and number of errors are recorded in ERROR.

Function:

D.C.CK is used to check if the ITEM retrieved by RD is either "DR" or "CR". "DEBIT", "D", and most common mispellings of "DEBIT" are also recognized as "DR". Similarly, "CREDIT", "C", and most common mispellings of "CREDIT" are recognized as "CR". If a "DR" or "CR" is found, D.C.CK stores the ITEM in DR.CR and returns a value of 0 if the ITEM is "DR", and 1 if the ITEM is "CR".

If ITEM is neither "DR" or "CR", D.C.CK flags a format error (FMT.ER), prints out an error message to the student, stores his answer in the output buffer and returns.

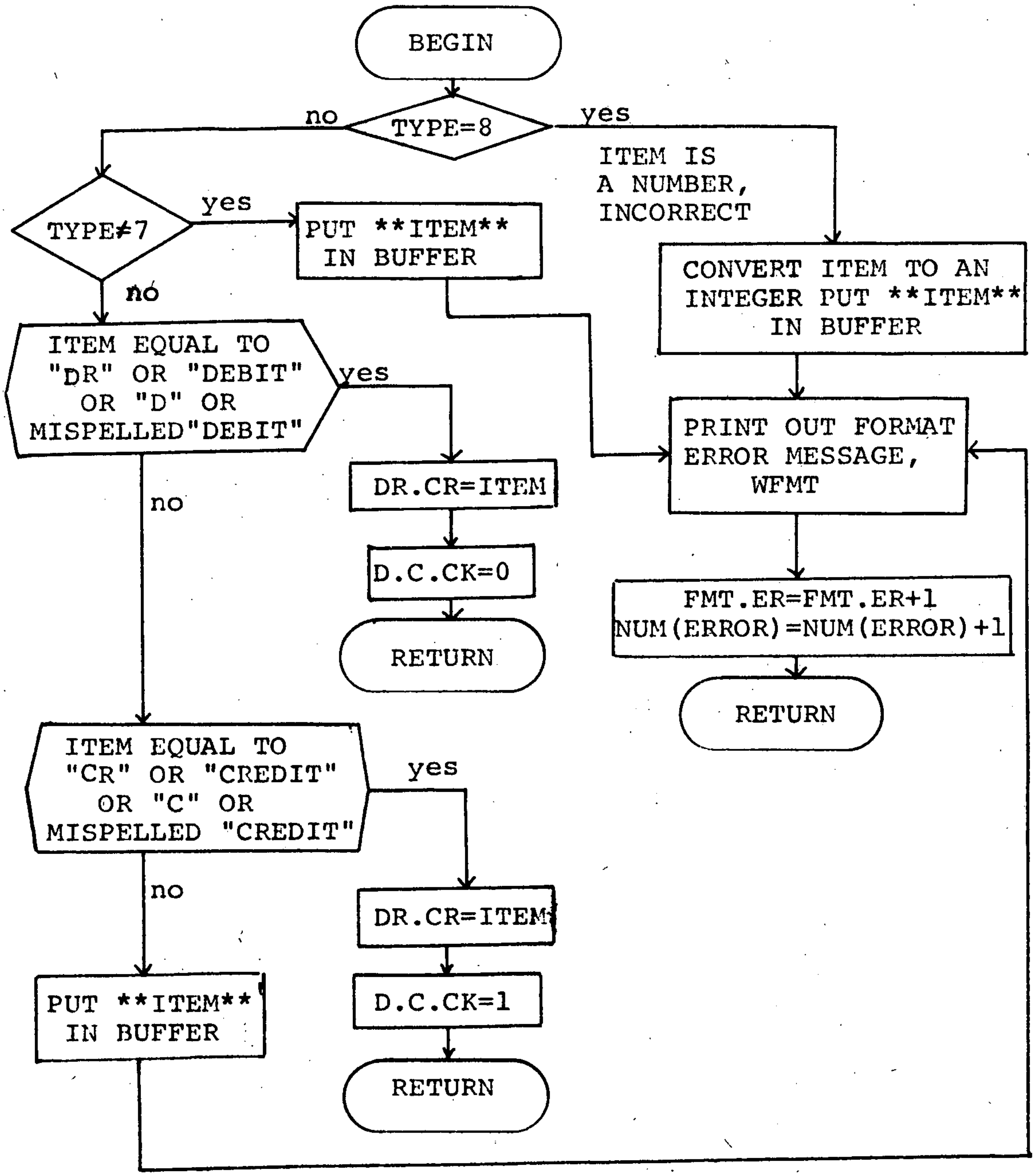
Called Procedures:

Name	Location
AEQL	AEQL ALGOL
ASC.C.	AIMSLB ALGOL
ASCINT	" "
ASCSAV	" "
CCOPY	" "
CFRET	" "
CPY.LN	CPY.LN ALGOL
CVTOIN	AIMSLB ALGOL
MISPEL	MISPEL ALGOL
NEWLIN	AIMSLB ALGOL
STOP	STOP ALGOL

WFMT WFMT ALGOL

Figure 30

Flow Chart for D.C.CK



<u>Procedure</u>: FILE

<u>Location</u>: FILE ALGOL

<u>Calling Sequence</u>: FILE(NAME1,ENT)

where

NAME1 Is a pointer to the student's first name.

ENT Is a pointer to the current question.

<u>Function</u>:

This procedure opens the files TUTORS STATUS and CLOSES STATUS which are used to maintain student statistics. TUTORS STATUS, currently used only to record the student's name, will be used to monitor the last question answered if a student does not complete the program at one sitting; this will make it possible for the student to re-enter the program where he left off.

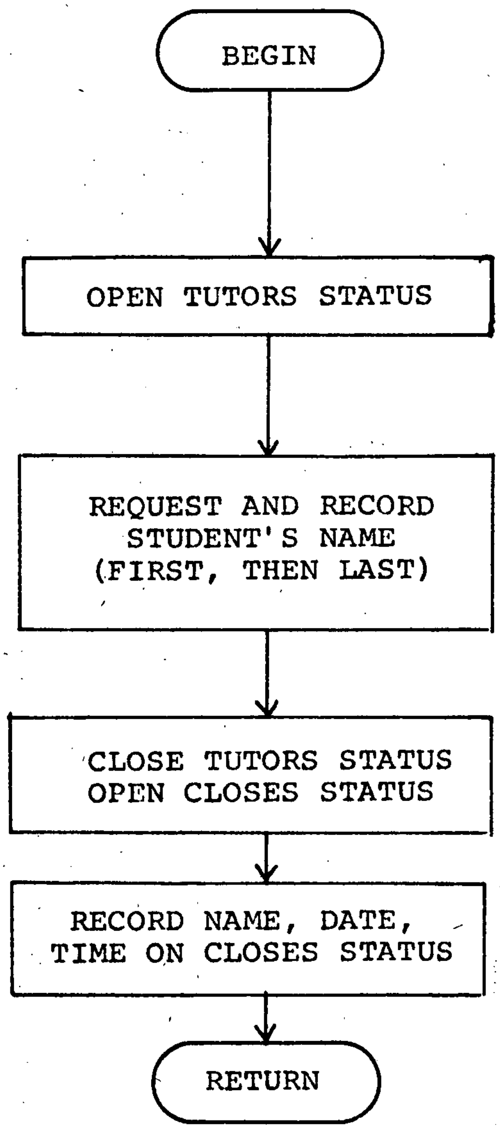
CLOSES STATUS is a complete monitor of the tutorial session. It records the date, time, and name of the student. In addition, it records every student response as well as an error code indicating the errors found in answers to questions.

<u>Called Procedures</u>:

Name	Location
ASC.C.	AIMSLB ALGOL
ASCDAT	" "
ASCLOS	" "
ASCOPN	" "
ASCOUT	" "
ASCSAV	" "
ASCTIM	" "
ASCWRP	" "
CPY.LN	CPY.LN ALGOL
NEWLIN	AIMSLB ALGOL
RD	SETBUF ALGOL

Figure 31

Flow Chart for FILE



Procedure: FLOCHT

Location: FLOCHT ALGOL

Calling Sequence: R FLOCHT

R DISPIC FLOCHT

Function:

FLOCHT draws the expenditure-asset-expense cycle picture. It also writes the picture on disk so that it may be quickly and easily accessed.

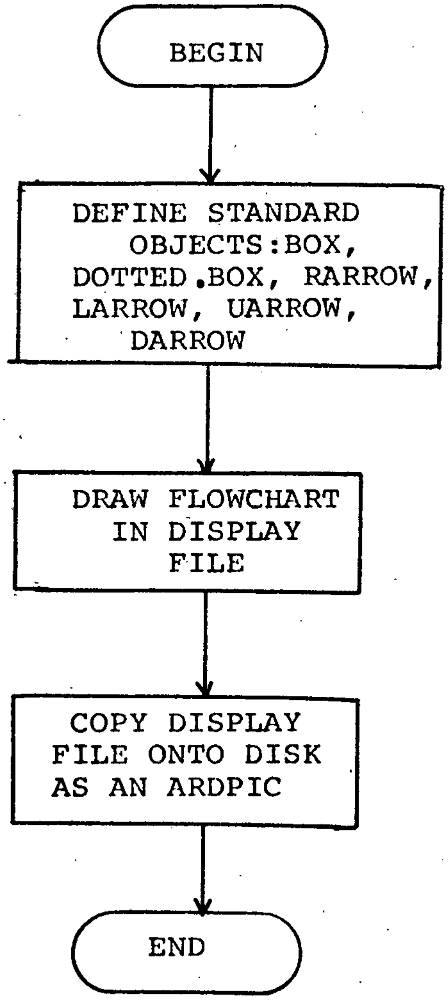
Called Procedures:

Name	Location
ADDOBJ	ABDS
COMBINE	"
CPYOBJ	"
DEFOBJ	"
DISKOB	"
DISKPIC	"
DOTTED	"
ENDOBJ	"
ERASER	"
INVIS	"
LIN	"
PLOT	"
SETPT	"
TEXT	"

D

Figure 32

Flow Chart for FLOCHT



Procedure: FWD

Location: FWD ALGOL

Calling Sequence: FWD()

Function:

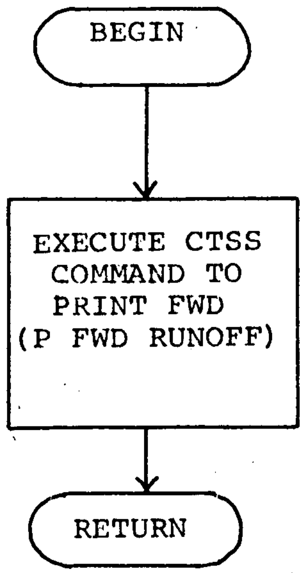
This procedure executes the CTSS command XECOM which causes the introduction contained in FWD RUNOFF to be output in a report form with both margins justified.

Called Procedures:

Name	Location
XECOM	CTSS
P	"

Figure 33

Flow Chart for FWD



Procedure: HELP

Location: HELP ALGOL

Calling Sequence: HELP(ACT.PTR,ENT)

where

ACT.PTR Is a pointer to the list of answer beads which have not been given as responses to the current question.

ENT Is a pointer to the current question.

Function:

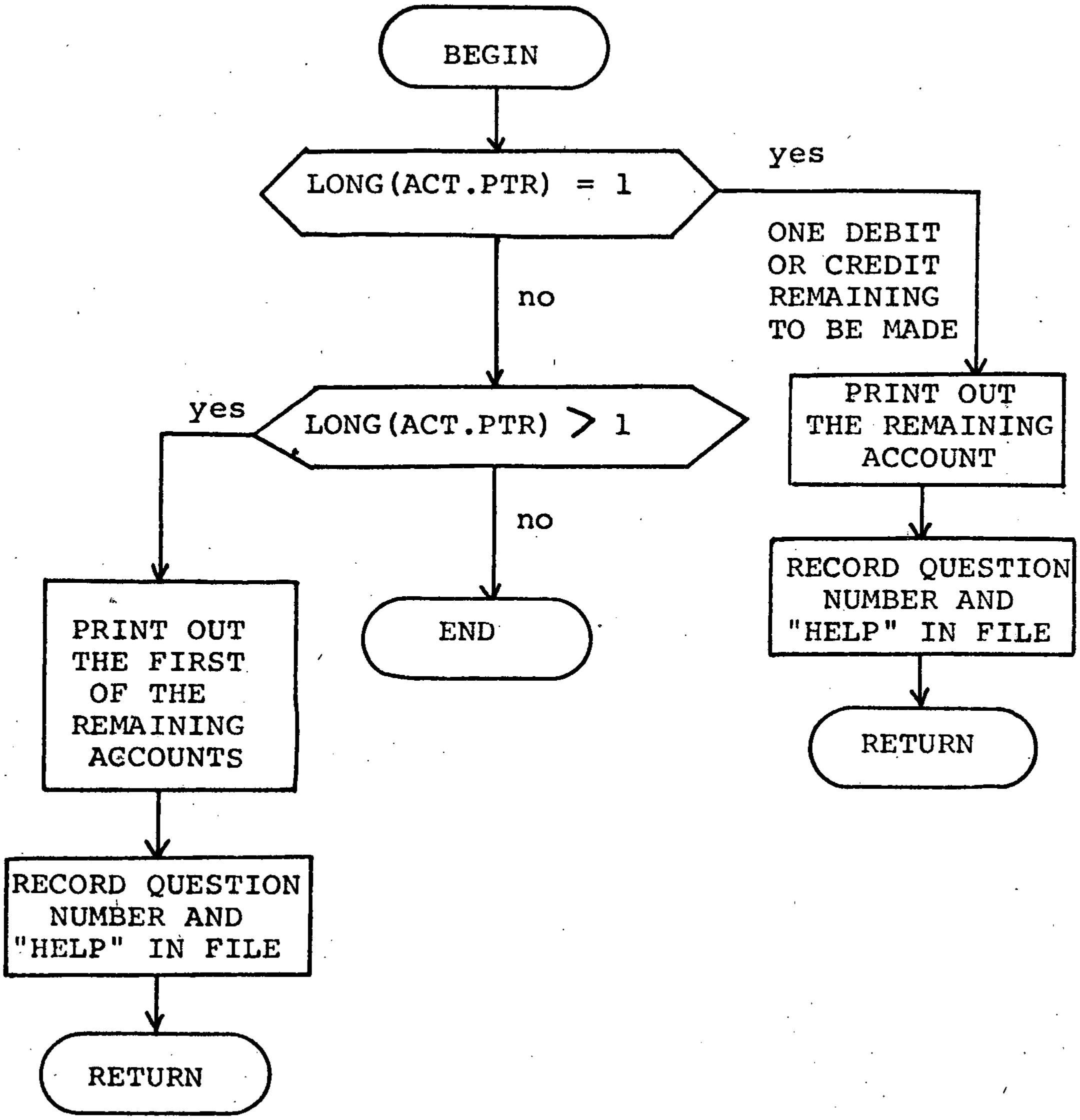
HELP prints out one of the remaining accounts in the answer, i.e. an account which the student has not yet given as part of an answer. HELP does not move the account from ACT.PTR to RT.PTR, but rather requires that the student type in the entry, which, if correct, causes the account to be moved to RT.PTR.

Called Procedures:

Name	Location	
ASC.C.	AIMSLB	ALGOL
ASCINT	"	"
ASCOUT	"	"
ASCWRP	"	"

Figure 34

Flow Chart for HELP



Procedure: LINCNT

Location: LINCNT ALGOL

Calling Sequence: LINCNT(NEED)

where

NEED Is an integer specifying the number of lines that are needed.

Function:

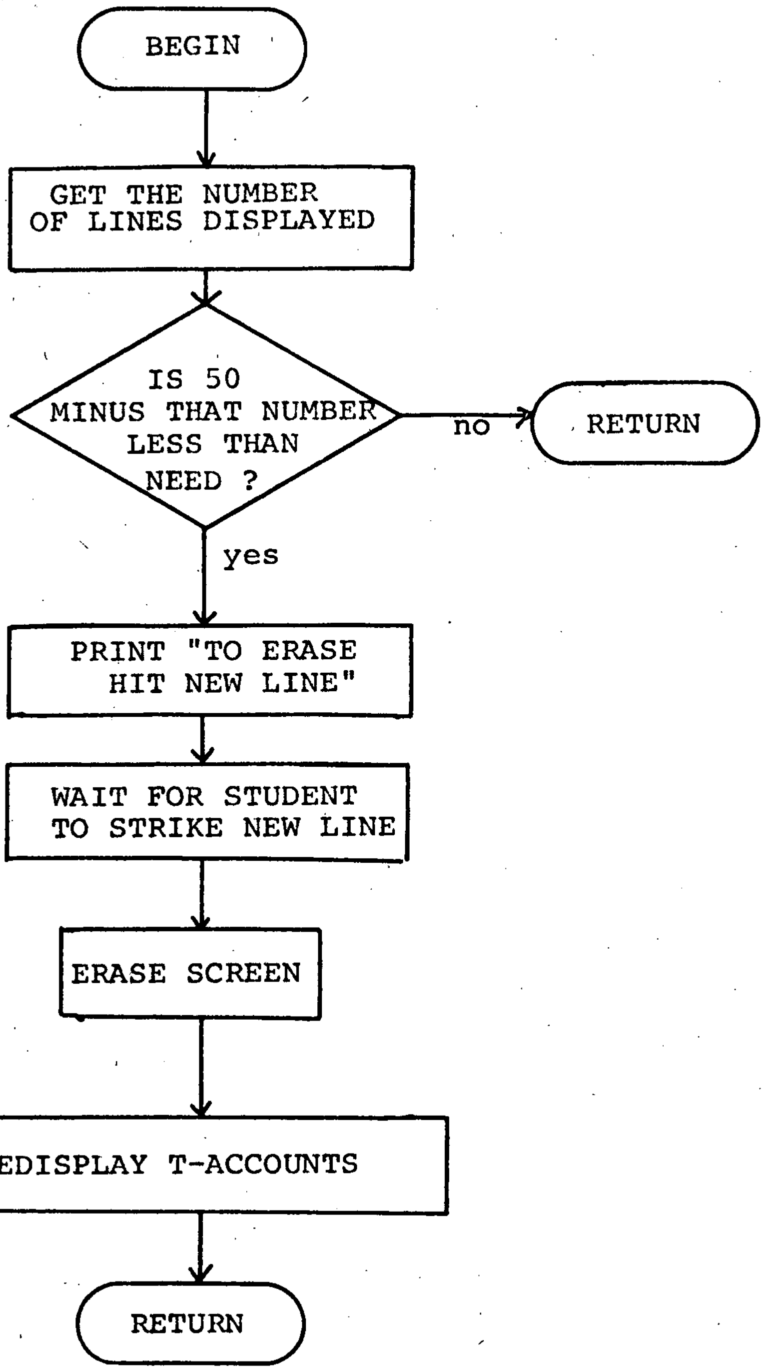
This procedure calls the CTSS supervisor to obtain the number of lines which have been displayed thus far. If the number of lines remaining, (i.e. fifty minus the number of lines displayed), is less than NEED, (the number of lines needed), the student is told to hit new line to erase the screen. When he hits new line, the screen is erased, and the top third, the T-accounts, is redrawn.

Called Procedures:

Name	Location
DISPLAY	ARDS
ERASER	"
GETP	"

Figure 35

Flow Chart for LINCNT



Procedure: MISPEL

Location: MISPEL ALGOL

Calling Sequence: MISPEL(STRNG1,STRNG2,N)

where

STRNG1 Is a pointer to the string of characters typed in by the student.

STRNG2 Is a pointer to the correct answer, also a string of characters.

N Is an integer specifying the minimum length of substring STRNG1 which must match a substring of STRNG2 in order for an acceptable match, i.e. if N=4, a substring of length 4 of STRNG1 must match a substring of STRNG2 of equal length to have an acceptable match.

Function:

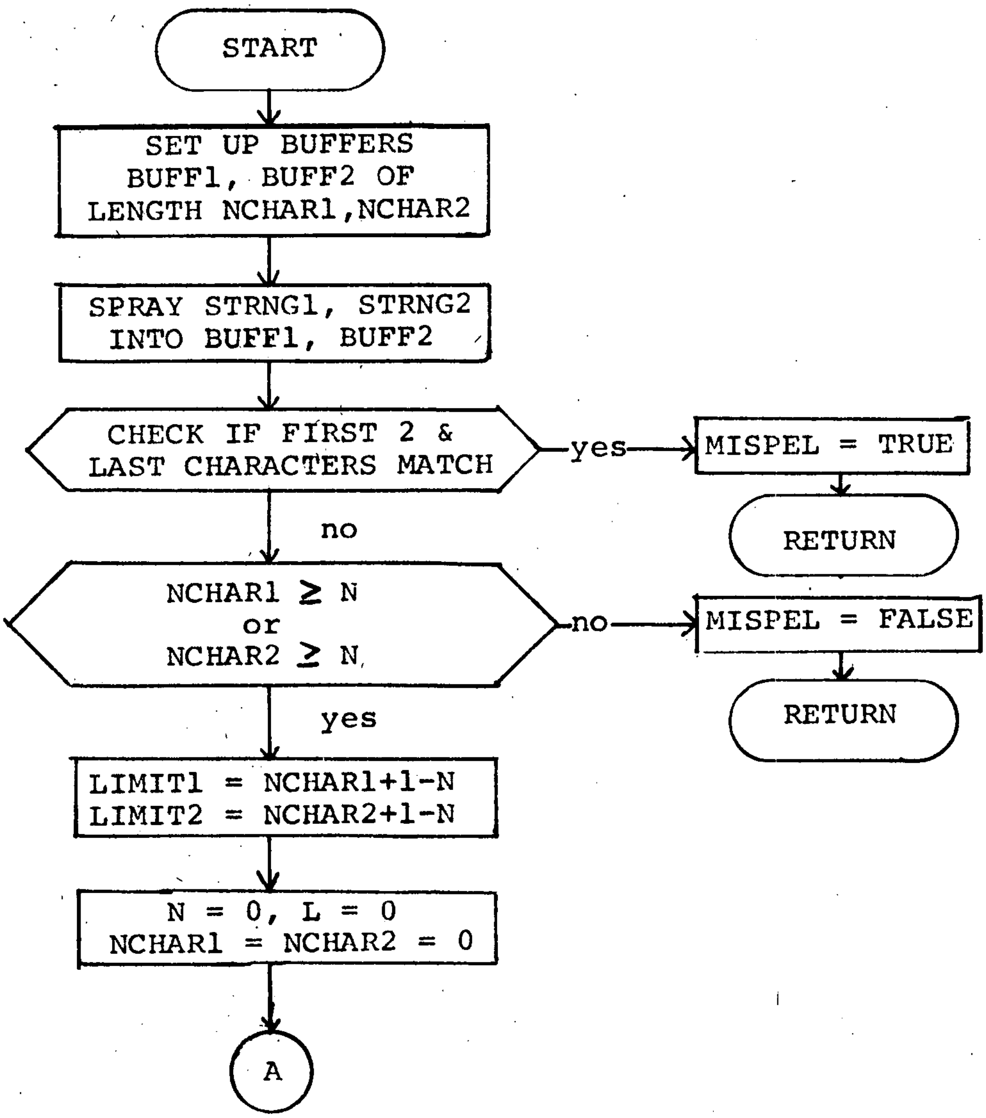
MISPEL attempts to match the character string pointed to by STRNG1 with the string pointed to by STRNG2. It uses two rules to determine if STRNG1 matches STRNG2. Initially, the first two and last characters of the strings are checked to see if they match. If they do, MISPEL returns TRUE; if they do not, MISPEL then checks to see if a substring of STRNG1 matches a substring of STRNG2 of equal length. It returns TRUE if such a match exists, and FALSE otherwise.

Called Procedures:

Name	Location
SPRAY9	AIMSLB ALGOL
FRET	" "
FREZ	" "

Figure 36

Flow Chart for MISPEL



(See page 150)

Figure 36 continued

From page 149

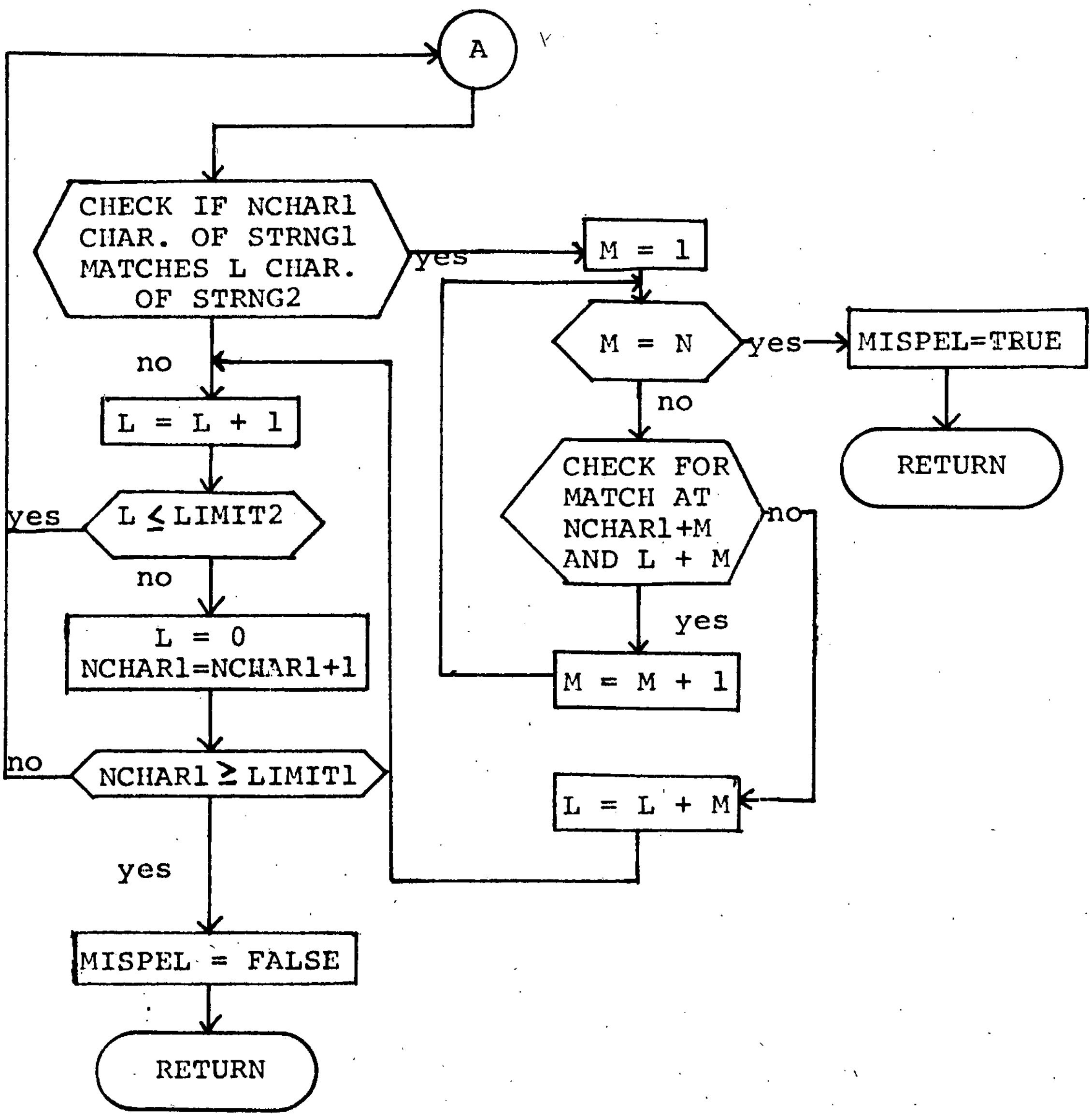
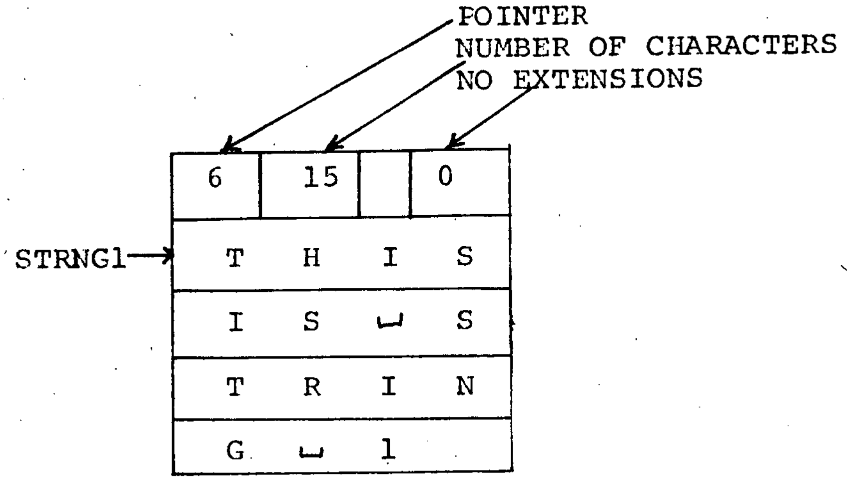
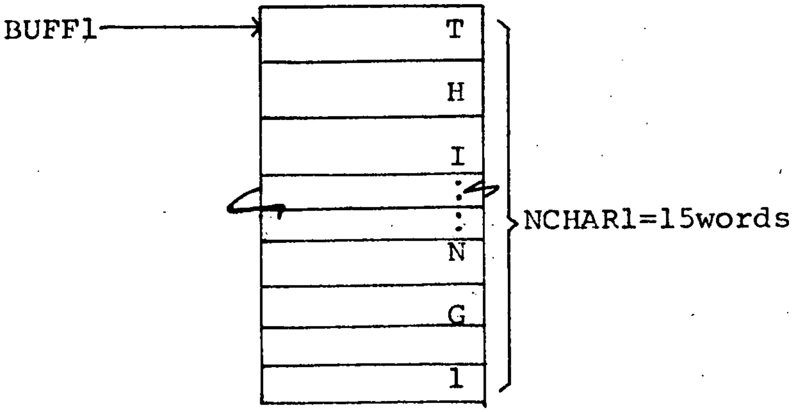


Figure 37

DETAIL ON CHARACTER STRINGS AND SPRAY9



After STRNG1 is sprayed into BUFF1 it looks like



Later NCHAR1 is used as a pointer to the current character for STRNG1

Symbollically,

BUFF1 ⟶ THIS␣IS␣STRNG 1
↑ NCHAR1

(The NCHAR1th character of STRNG1 is S)

Procedure: MLTPST

Location: MLTPST ALGOL

Calling Sequence: MLTPST(ENT)

where

ENT Is a pointer to the current question.

Function:

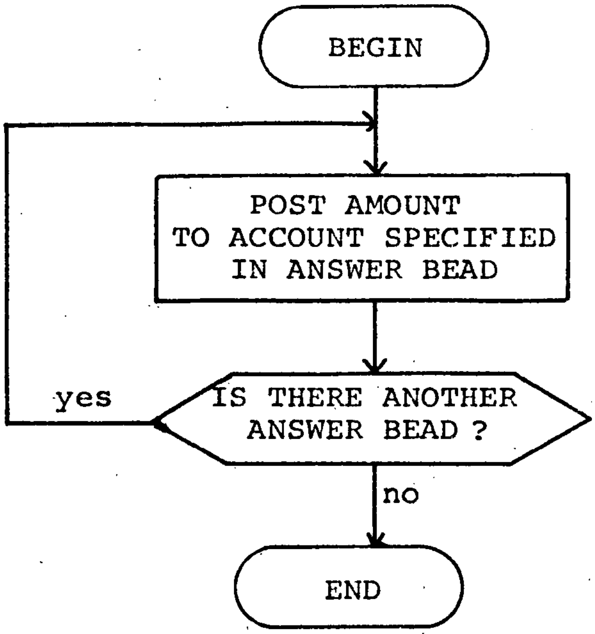
Procedure MLTPST posts the answer to the various T-accounts involved. It makes calls on POST to do the actual posting.

Called Procedures:

Name	Location
POST	POST ALGOL

Figure 38

Flow Chart for MLTPST



Procedure: OPWORD

Location: D.C.CK ALGOL

Calling Sequence: OPWORD()

Function:

OPWORD sets up .C. pointers to the character string representations of D,DR,DEBIT,C,CR, and CREDIT. These pointers are used with the procedure AEQL in procedure D.C.CK to determine if the item typed by the student is a debit or a credit.

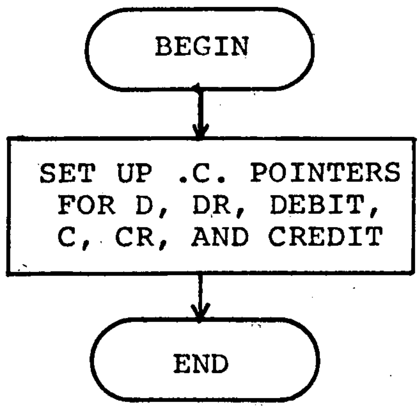
Because of AED limitations OPWORD must reside in the same file as D.C.CK, i.e. D.C.CK ALGOL, in order for the pointers to be known in procedure D.C.CK.

Called Procedures:

Name	Location
ASC.C.	AIMSLB ALGOL
ASCSAV	" "

Figure 39

Flow Chart for OPWORD



Procedure: PIC1

Location: PIC1 ALGOL

Calling Sequence: PIC1(A)

where

A Is an array whose elements are pointers to the various T-accounts.

Function:

This procedure plots the T-accounts in the display file. PIC1 first defines two standard T-accounts, the long form (i.e. Cash), and the short form (i.e. Wages). It then sets the beam to the appropriate position on the screen for the account, and then plots one of the two standard T-accounts; PIC1 also plots the name of the account.

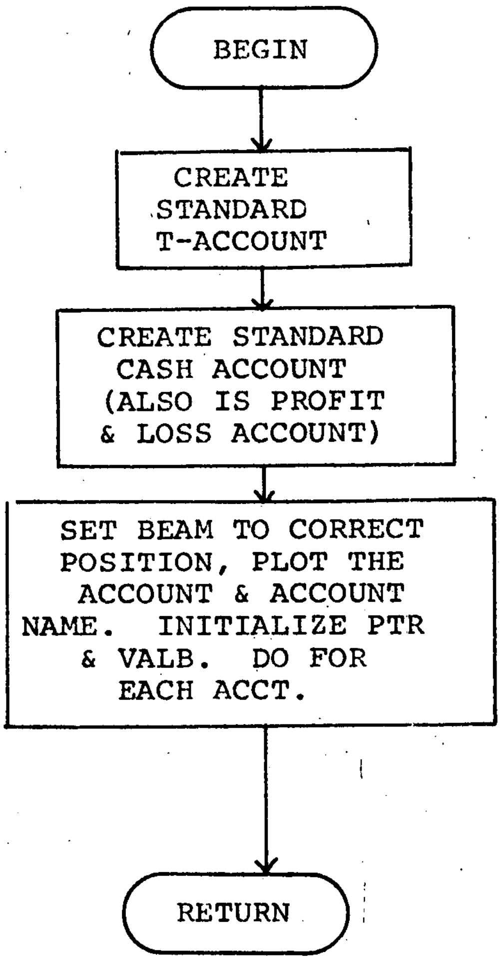
This procedure also initializes PTR and VALB of each account bead (see the description of the account bead structure.). It sets PTR to the display file pointer for the account, and VALB equal to the .C. string spelling of the account name.

Called Procedures:

Name	Location
ADDOBJ	ARDS
DEFOBJ	"
ENDOBJ	"
INVIS	"
LIN	"
PLOT	"
SETPT	"
TEXT	"

Figure 40

Flow Chart for PIC1



Procedure: POST

Location: POST ALGOL

Calling Sequence: POST(ACCT,D.C,AMT1)

where

ACCT Is a pointer to an account.

D.C Is an integer. Acceptable values are:

0 indicates a debit entry.

1 indicates a credit entry.

3 indicates that the account is to be underlined and the balance brought forward.

4 indicates that the account is to be underlined (twice).

AMT1 Is the amount to be debited or credited.

Function:

POST is the procedure that handles all entries to the T-accounts. It makes all debit and credit entries, and also underscores the accounts when required.

If D.C is either 0 or 1, the procedure posts the amount specified by AMT1 to the account specified by ACCT. The entry is flagged with an arrow to make it easy for the student to spot the new entry.

If D.C is either 3 or 4, POST underlines the account specified. If D.C is 3, POST enters a single underline and posts the balance forward by calling itself (POST is a recursive procedure) to do the posting. For D.C equal to 4 POST underlines the account specified twice.

This process of making entries to the accounts and underlining them is facilitated by the maintenance of certain "vital statistics" for each account, e.g. X,Y coordinates, etc. (see Bead Description on p. xxx). When POST is called to enter an amount, it merely accesses the necessary statistics from the bead maintained for ACCT. It makes the entry on the screen, updates the statistics which have changed, and returns.

Called Procedures:

Name	Location
ASC.C.	AIMSLB ALGOL
ASCARD	" "
ASCINT	" "

ASCSAV	"	"
DISPLAY	ARDS	
FREZ	AIMSLB	ALGOL
INVIS	ARDS	
PLOT	"	
RMV	"	
SETPT	"	

Figure 41

Flow Chart for POST

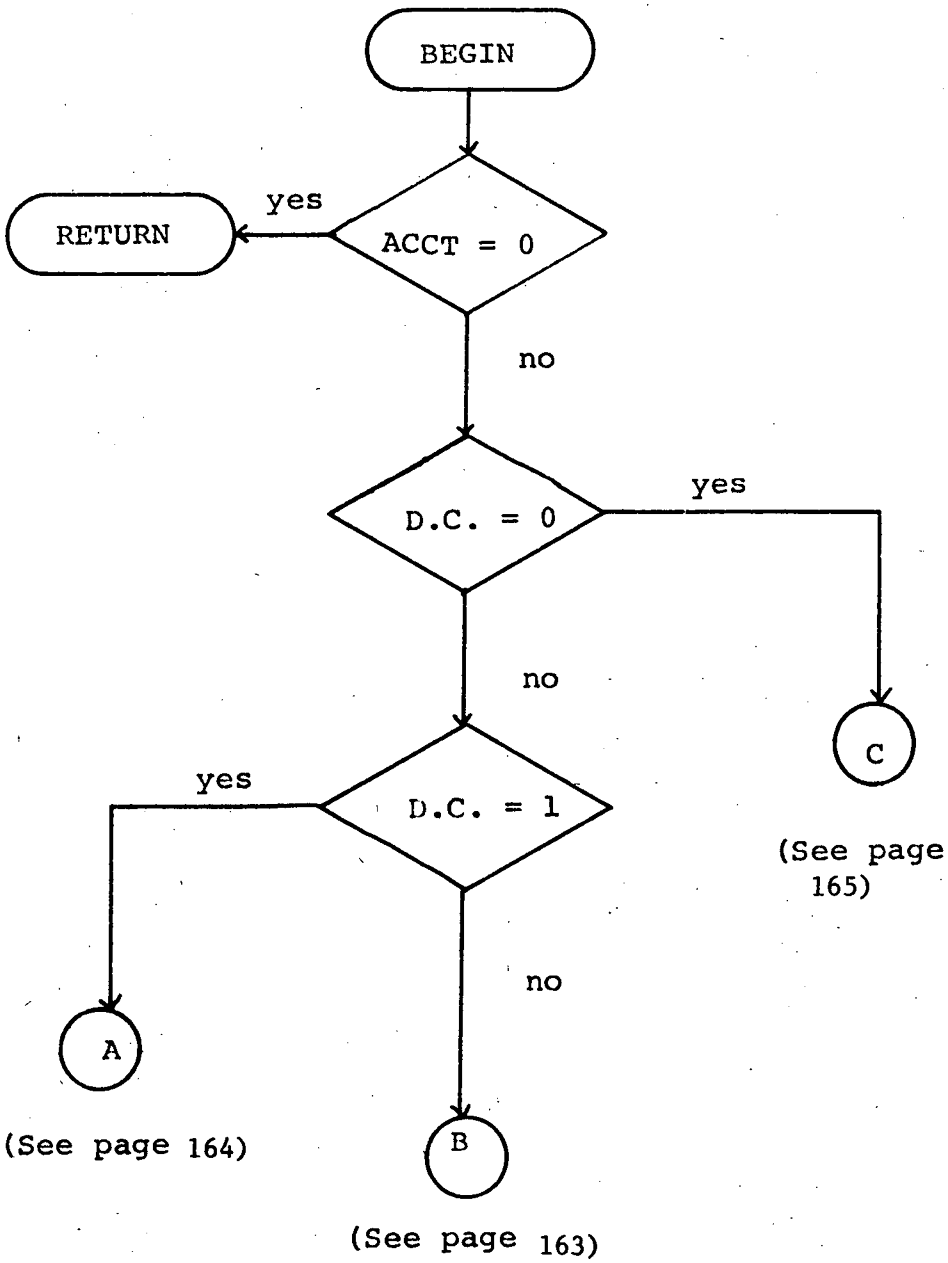


Figure 41 continued

FROM PAGE 162

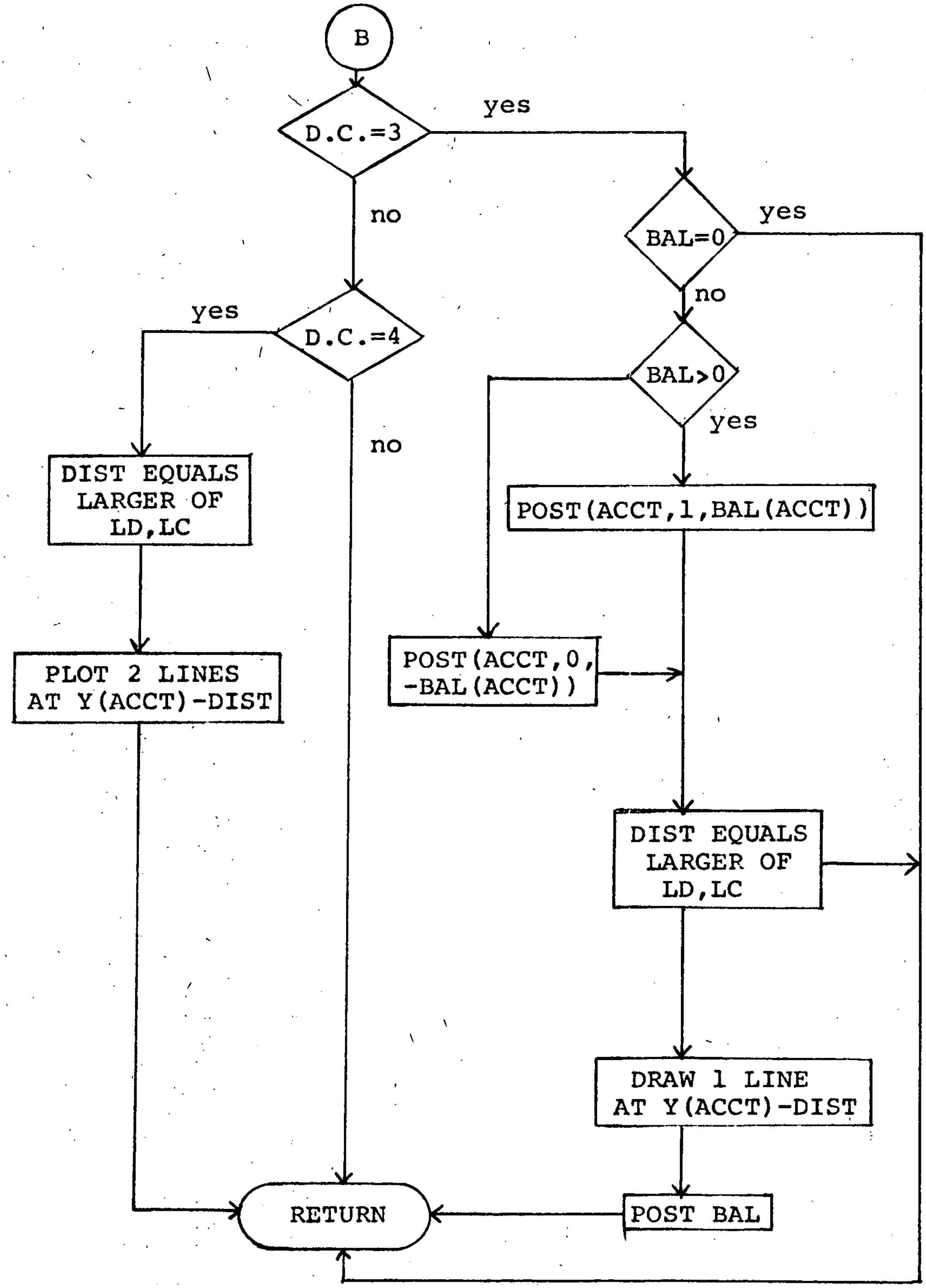


Figure 41 continued

FROM PAGE 162

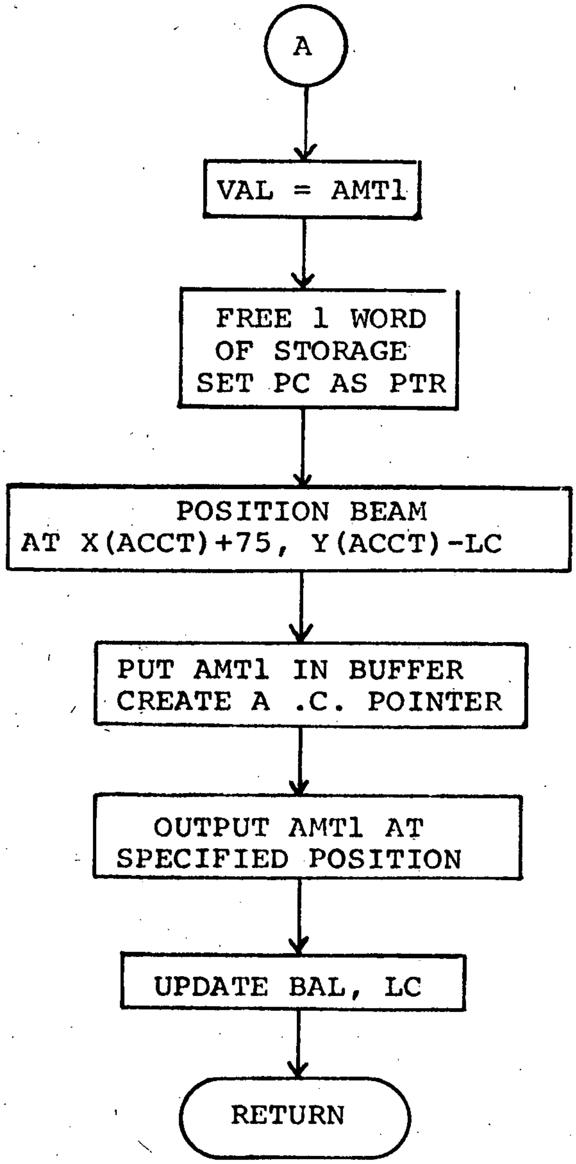
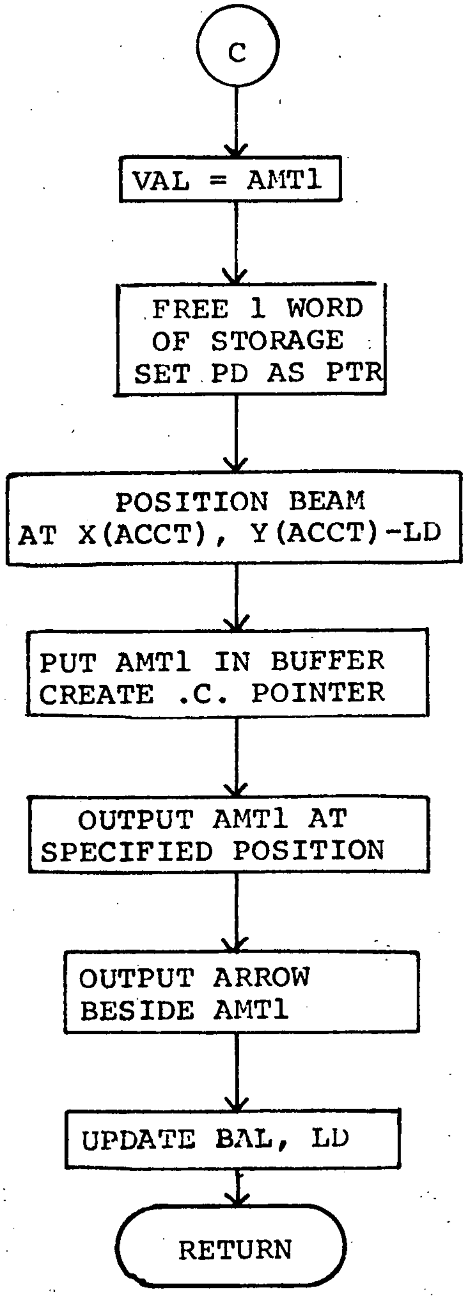


Figure 41 continued

FROM PAGE 162



Procedure: POST.1

Location: POST.1 ALGOL

Calling Sequence: POST.(A,QPTR)

where

A	Is an array whose elements are pointers to the various T-accounts.
QPTR	Is an array whose elements are pointers to the various questions.

Function:

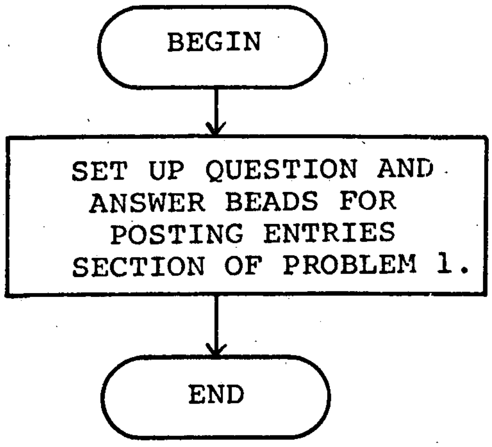
This procedure sets up the question and answer beads for the posting entries section of problem 1.

Called Procedures:

Name	Location
FREZ	AIMSLB ALGOL

Figure 42

Flow Chart for POST.1



Procedure: PROB1

Location: PROB1 ALGOL

Calling Sequence: PROB1(A,QPTR,N)

where

A Is an array whose elements are pointers to the various T-accounts.

QPTR Is an array whose elements are pointers to the various questions.

N Is an integer whose value may be 1 or 2. If N equals 1, PROB1 initializes problem 1. If N equals 2, PROB1 posts the starting balances to the T-accounts.

Function:

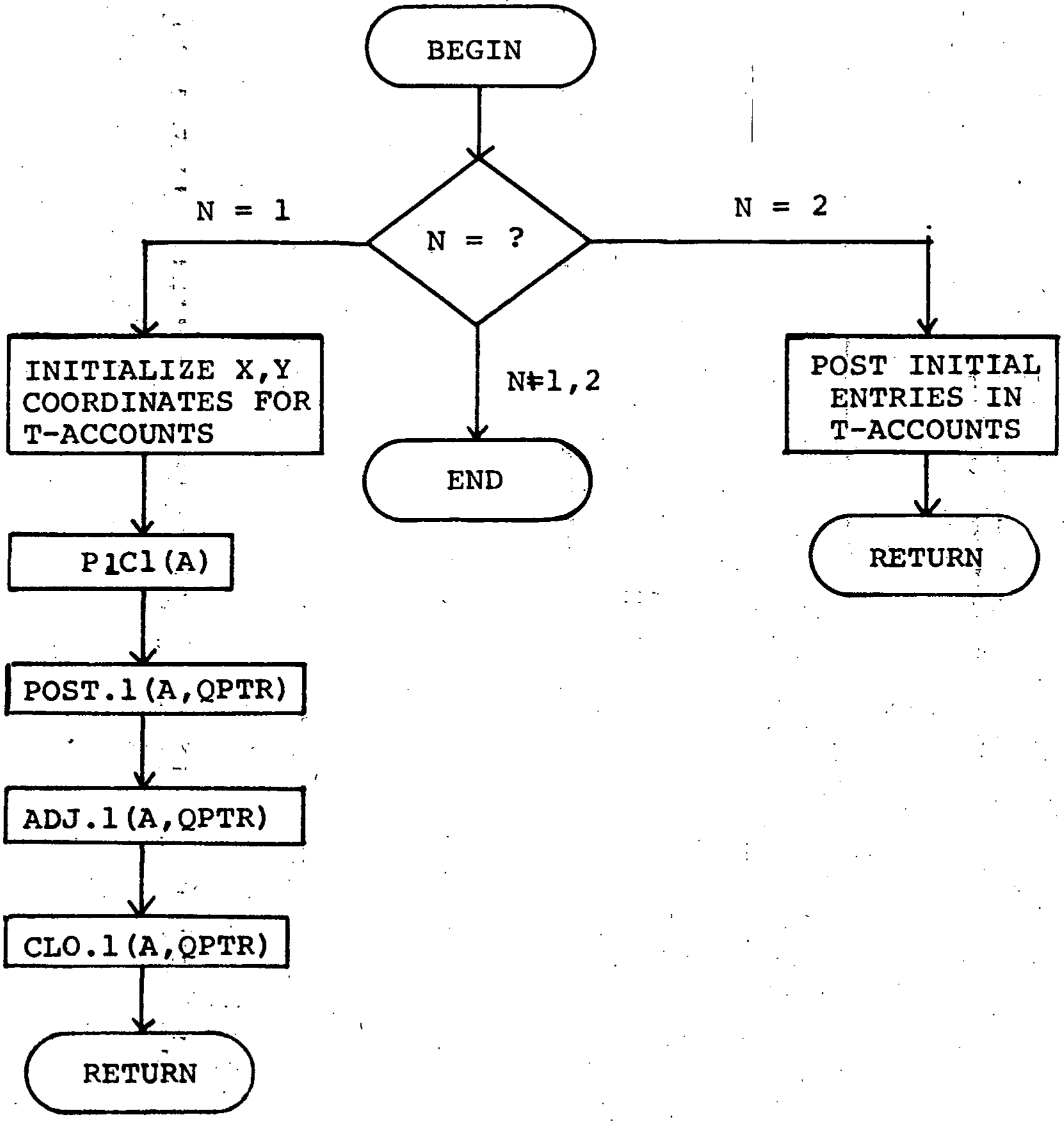
This procedure is the main subroutine for problem 1. It initializes variables and calls the necessary subroutines to set up problem 1. It is called twice from CLOSE. On the first call, PROB1 initializes X,Y coordinates for the T-accounts and sets up the question and answer heads and returns. The second time PROB1 is called it posts initial balances to the T-accounts and returns.

Called Procedures:

Name	Location
ADJ.1	ADJ.1 ALGOL
CLO.1	CLO.1 ALGOL
PIC1	PIC1 ALGOL
POST	POST ALGOL
POST.1	POST.1 ALGOL

Figure 43

Flow Chart for PROB1



Procedure: RD

Location: SETBUF ALGOL

Calling Sequence: RD(TYPE,STR,ENT)

where

TYPE Is an integer indicating the type of the item, i.e. TYPE=8 indicates that the item is an integer (see AIMSLB documentation, <30>, for list of item types).

STR Is a pointer to the item.

ENT Is a pointer to the current question.

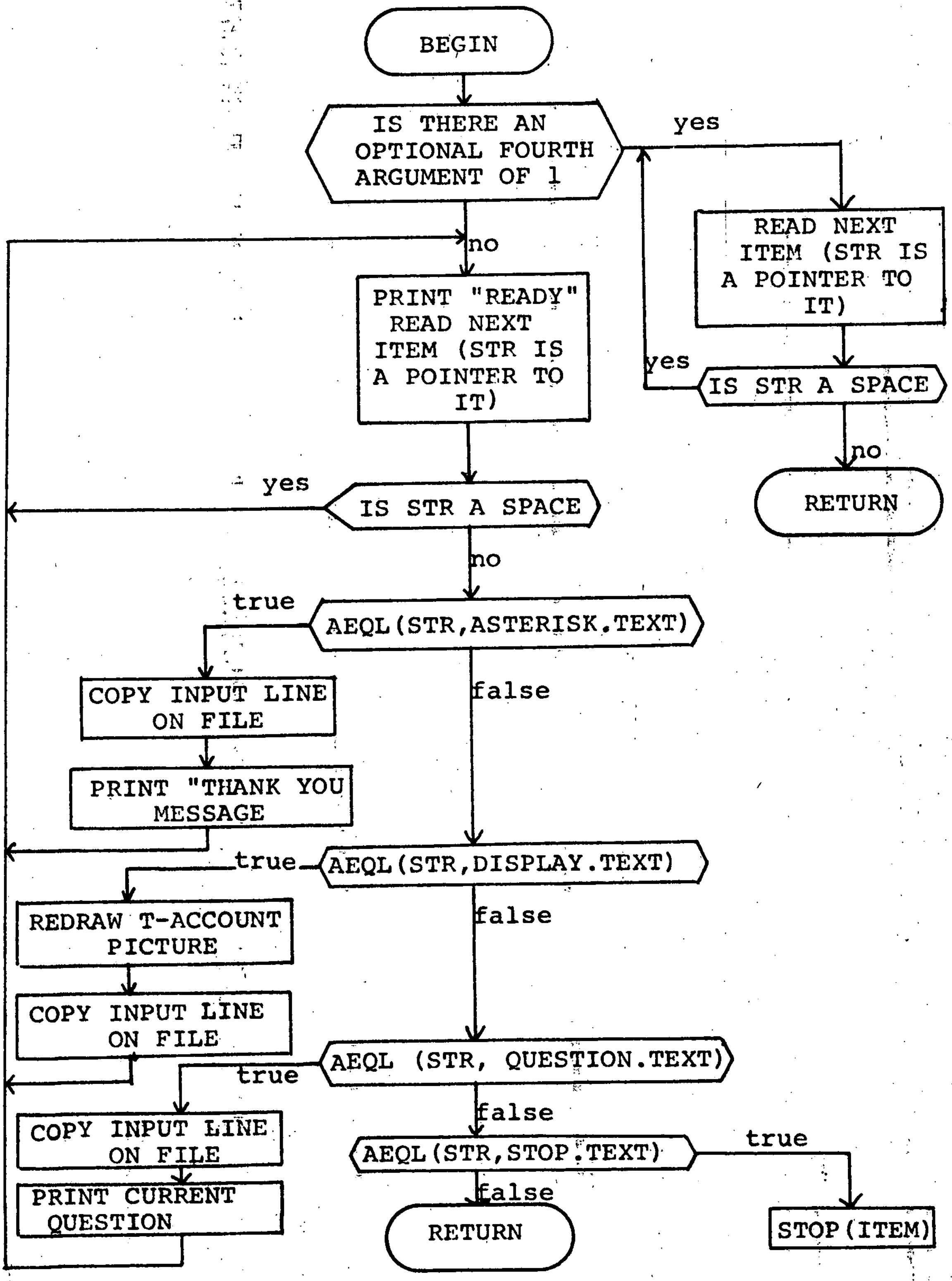
Function:

The procedure RD reads in the next item from the console, ignoring spaces. In additicn, it checks for certain control words before returning the item. It checks for "STOP", "DISPLAY", "QUESTION", and "*". If RD finds one of these control words, it handles the request and then seeks and returns a new item.

Called Procedures:

Name	Location
AEQL	AEQL ALGOL
ASC.C.	AIMSLB ALGOL
ASCINT	" "
ASCOUT	" "
ASCWRF	" "
CPY.LN	CPY.LN ALGOL
DISPLAY	ABDS
ISARGV	AED
NEWLIN	AIMSLB ALGOL
NXTITM	" "
STOP	STOP ALGOL

Figure 53
Flow Chart for RD



Procedure: REPLIN

Location: REPORT ALGOL

Calling Sequence:

REPLIN(ACCT1,TAB10,TAB11,ACCT2,TAB20,TAB21)

where -

ACCT1 Is the first account on the report line and, for the current report generator, is always a balance sheet account.

TAB10 Is the character position where ACCT1 is printed.

TAB11 Is the character position where the balance of ACCT1 is printed.

ACCT2 Is the second account on the report line; currently, ACCT2 is always an income or an expense account.

TAB20 Is the character position where the name of ACCT2 is printed.

TAB21 Is the character position where the balance of ACCT2 is printed.

Function:

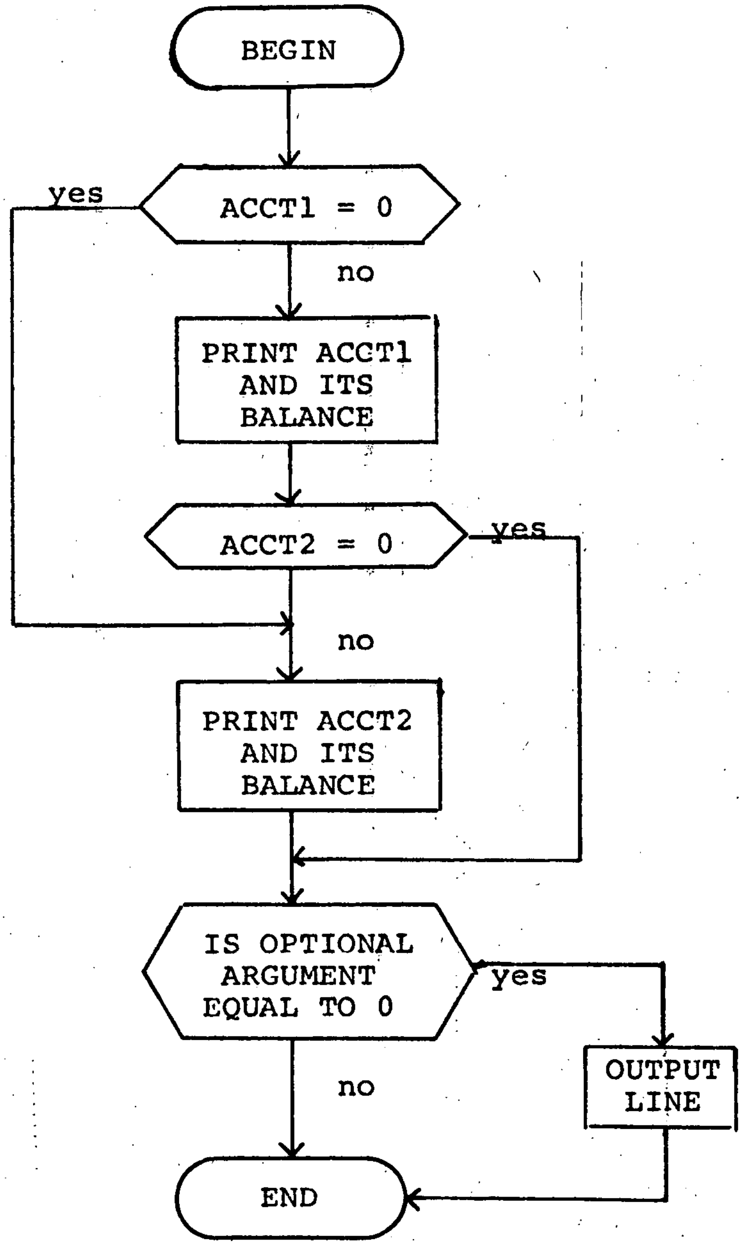
Procedure REPLIN prints one line of the financial reports. The accounts and positions at which to print the accounts and amounts are specified by argument values. Normally, two accounts are printed per line; however, it is also possible to print only one account, either ACCT1 or ACCT2, per line. To print only one account on a line, the account position bypassed should be zero, i.e. ACCT1=0 will print only ACCT2.

Called Procedures:

Name	Location	
ASC.C.	AIMSLB	ALGOL
ASCINT	"	"
ASCOUT	"	"
ASCTAB	"	"
ISARGV	AED	

Figure 45

Flow Chart for REPLIN



Procedure: REPORT

Location: REPORT ALGOL

Calling Sequence: REPORT(A,I)

where

A — Is an array whose elements are pointers to the various T-accounts.

I — Is an integer indicating whether initial reports (I=1) or final reports are to be drawn up.

Function:

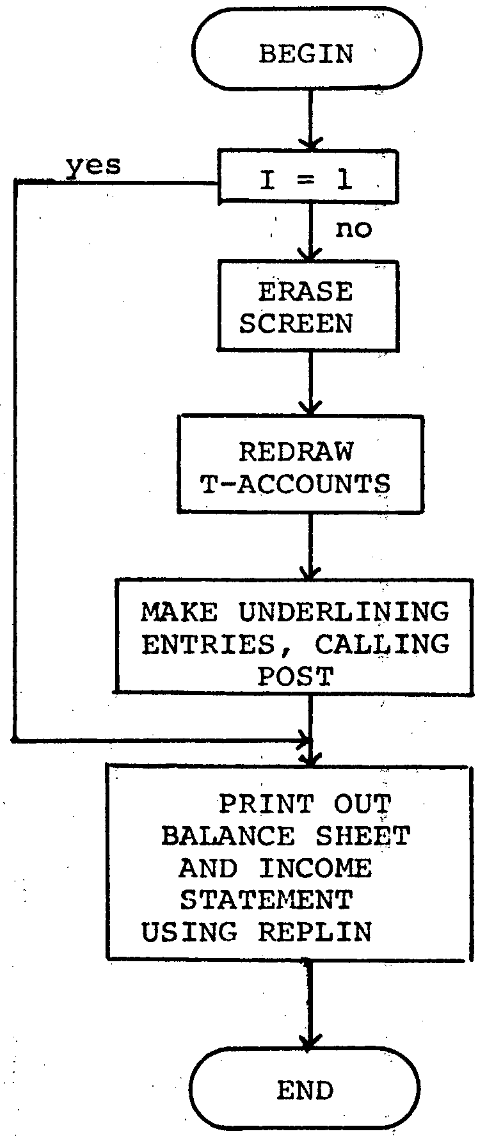
Procedure REPORT generates an income statement and a balance sheet. If I equals 1, REPORT does not alter the T-accounts, and gives an initial income statement and balance sheet. Otherwise, REPORT brings balances forward and makes underlining entries before printing the financial reports.

Called Procedures:

Name	Location
ASC.C.	AIMSLB ALGOL
ASCINT	" "
ASCOUT	" "
ASCTAB	" "
DISPLAY	ARDS
ERASER	"
POST	POST ALGOL
REPLIN	REPORT ALGOL

Figure 46

Flow Chart for REPORT



Procedure: SETBUF

Location: SETBUF ALGOL

Calling Sequence: SETBUF()

Function:

The procedure SETBUF establishes a buffer for console input. All characters typed in on the console are read into this buffer. SETBUF also sets up .C. pointers to the character string representations of "STOP", "DISPLAY", "QUESTION", and "*".

Because of AED limitations, SETBUF must reside in the same file as procedure RD, which will use the .C. pointers. The .C. pointers are established in SETBUF rather than RD, where they are used, because SETBUF is called once, while RD is called many times.

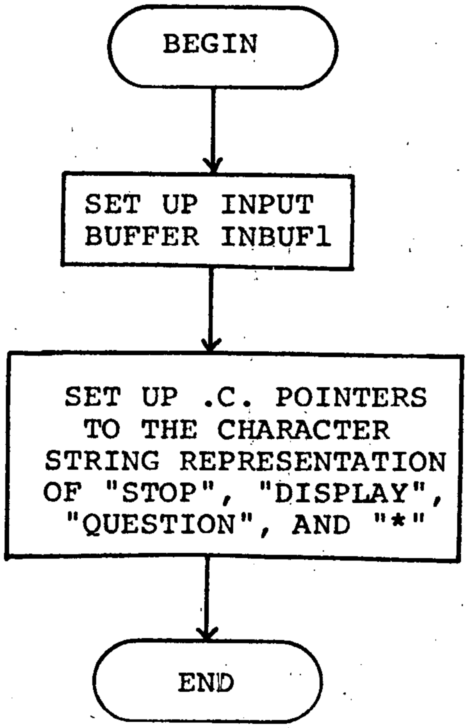
Called Procedures:

Name	Location
ASC.C.	AIMSLB ALGOL
ASCEK	" "
ASCHAR	" "

ASCONS	"	"
ASCSAV	"	"
FLXFMT	"	"
GENRB	"	"
OPNFIL	"	"

Figure 47

Flow Chart for SETBUF



Procedure: SETUP

Location: SETUP ALGOL

Calling Sequence: .INSERT ALGOL

Function:

SETUP is not really a procedure, but rather an insert file. It contains the variable and component declarations that are used throughout CLOSE. It is inserted at the beginning of every major procedure.

Called Procedures: No procedures called.

Procedure: SKIP

Location: SKIP ALGOL

Calling Sequence: SKIP(ENT)

where

ENT Is a pointer to the current question.

Function:

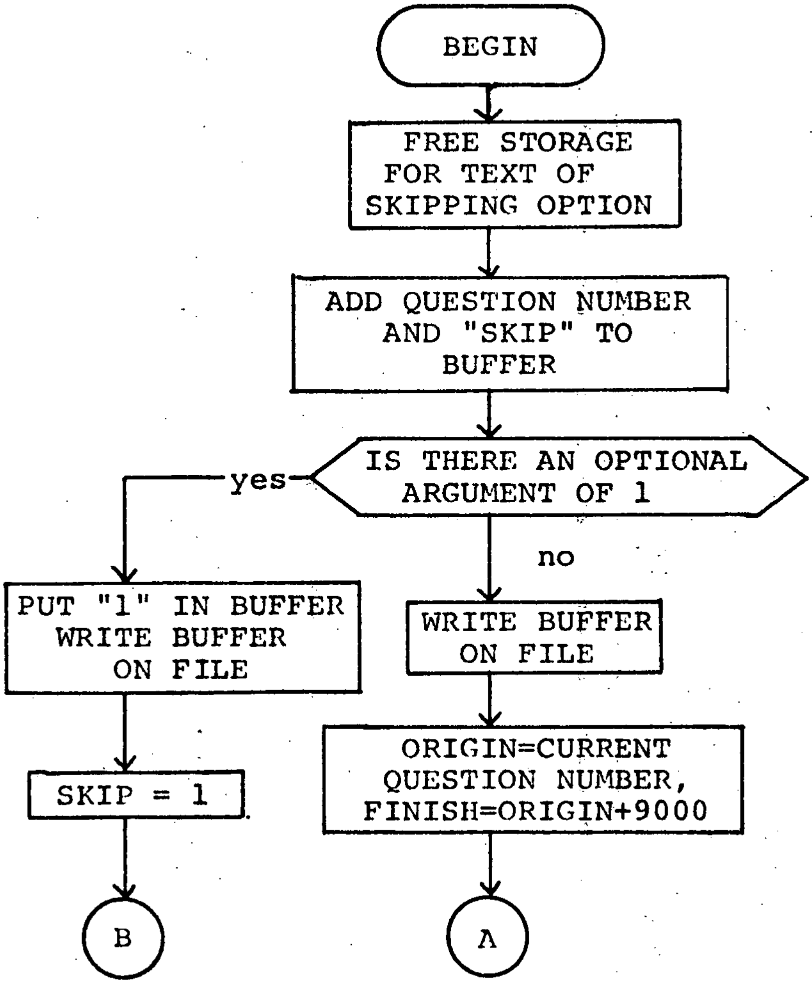
SKIP allows the student to skip to the next question or to the next section. SKIP, when called, prints out the three types of skipping that are available. They are: a) Print questions and answers while skipping to the next section, b) Just skip to the next section, and c) Skip to the next question. In addition, SKIP may be given an optional argument of 1 which suppresses printing of the options and automatically skips to the next question.

Called Procedures:

Name	Location
ASC.C.	AIMSLB ALGOL
ASCINT	" "
ASCOUT	" "
ASCWRP	" "
CHOOSE	CHOOSE ALGOL
CPY.LN	CPY.LN ALGOL
CVTOIN	AIMSLB ALGOL
FRET	" "
FREZ	" "
LINCNT	LINCNT ALGOL
MLTPST	MLTPST ALGOL
NEWLIN	AIMSLB ALGOL
RD	SETBUF ALGOL

Figure 48

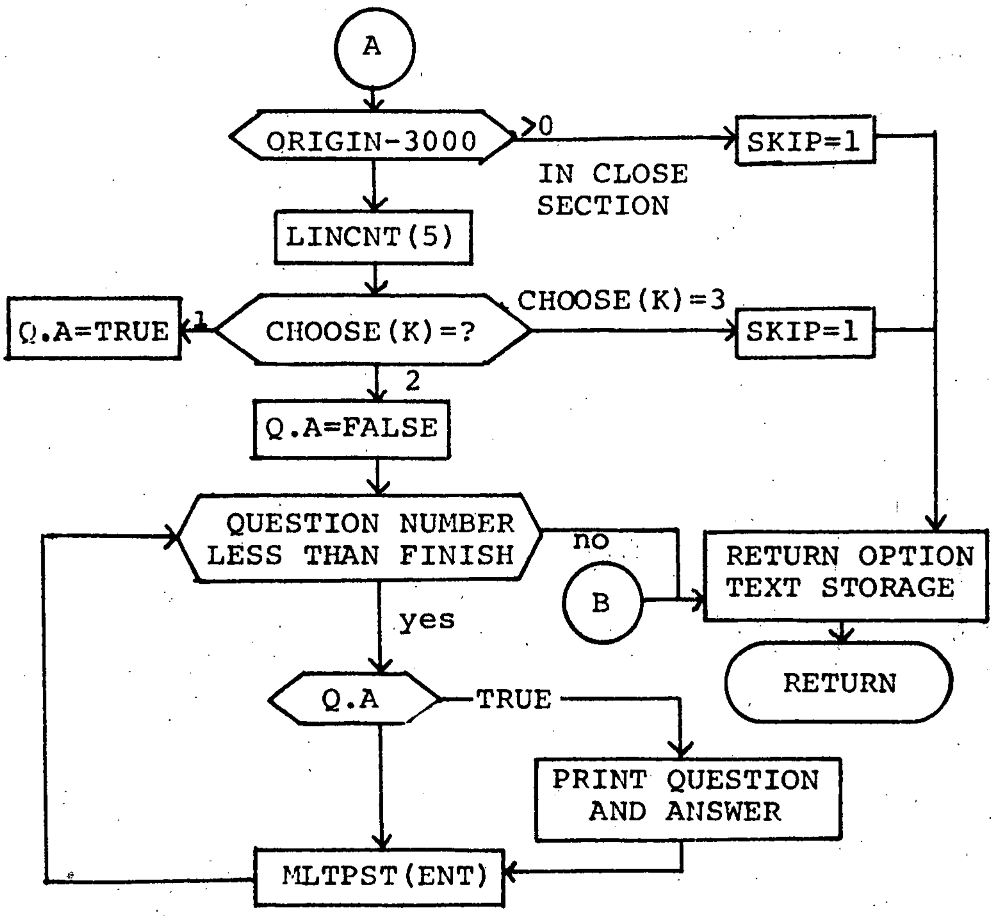
Flow Chart for SKIP



(See page 187)

Figure 48 continued

FROM PAGE 186



Procedure: STOP

Location: STOP ALGOL

Calling Sequence: STOP(ITEM)

where

ITEM Is a pointer to a string containing the characters END or STOP depending on where STOP was called.

Function:

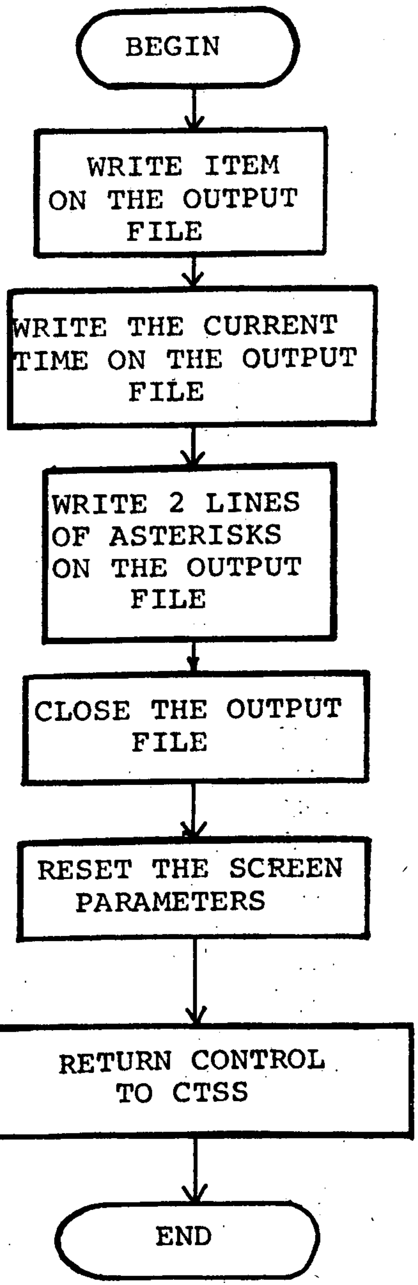
STOP is used to terminate the program. It adds to the output file, CLOSES STATUS, the word STOP or END depending on whether the program ended normally or was stopped before it ended. It also adds the current time to the file as well as two lines of asterisks to improve legibility, and then closes the file. In addition, STOP resets the screen parameters to their normal settings. STOP then returns control to CTSS.

Called Procedures:

Name	Location	
ASC.C.	AIMSLB	ALGOL
ASCLOS	"	"
ASCTIM	"	"
ASCWRF	"	"
SETPRM	ABDS	

Figure 49

Flow Chart for STOP



Procedure: TITLE

Location: TITLE ALGOL

Calling Sequence: TITLE()

Function:

This procedure erases the screen, outputs "SLOAN SCHOOL OF MANAGEMENT", "Accounting Program - Version SSM3 of 29 May 1970", the current date, and "CLOCK TIME= the current time".

It also outputs an appropriate greeting: GOOD MORNING (0000-1200 hours), GOOD AFTERNOON (1200-1700), and GOOD EVENING (1700-2400).

TITLE then waits for the student to hit new line, signifying that he is ready to continue, and erases the screen when he has done so.

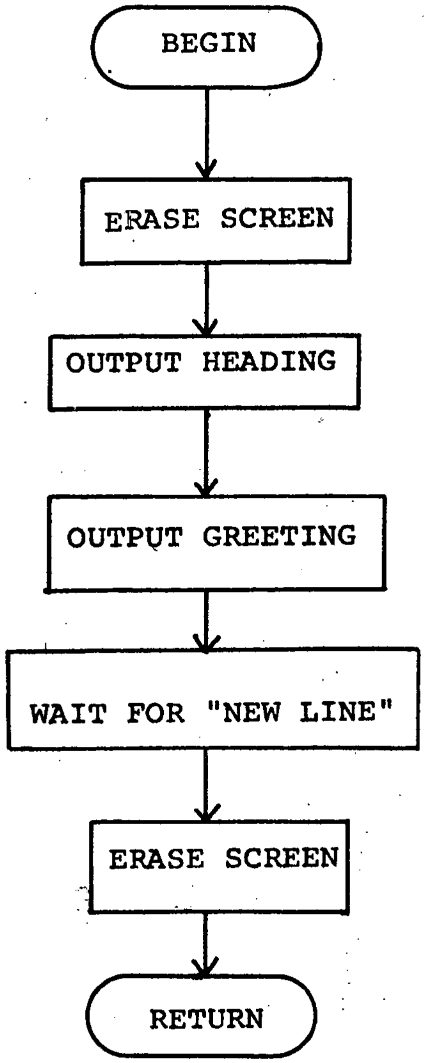
Called Procedures:

Name	Location
ASCTAB	AIMSLB ALGOL
ASC.C.	AIMSLB ALGOL
ASCDAT	" "

ASCOUT	"	"
ASCTAB	"	"
ASCTIM	"	"
ERASER	ARDS	
GOODAY	AIMSLB	ALGOL
RDFLX	CTSS	

Figure 50

Flow Chart for TITLE



Procedure: WFMT

Location: WFMT ALGOL

Calling Sequence: WFMT()

Function:

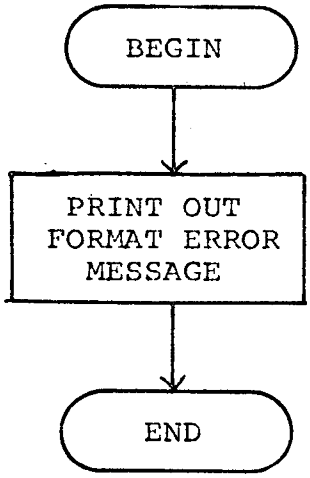
WFMT is called from D.C.CK to print out a format error message.

Called Procedures:

Name	Location
ASC.C.	AIMSLB ALGOL
ASCOUT	" "

Figure 51

Flow Chart for WFMT



Procedure: WORDS

Location: CHECK ALGOL

Calling Sequence: WORDS()

Function:

WORDS sets up .C. pointers to the character string representations of "SKIP" and "HELP". These pointers are used with the procedure AEQL in procedure CHECK to determine if the item typed by the student is "HELP" or "SKIP".

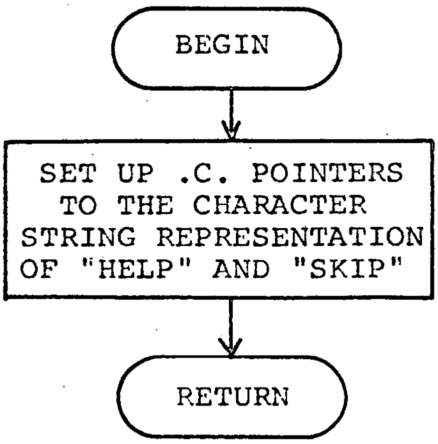
Because of AED limitations, WORDS must reside in CHECK ALGOL in order for the pointers to be known in procedure CHECK.

Called Procedures:

Name	Location
ASC.C.	AIMSLB ALGOL
ASCSAV	" "

Figure 52

Flow Chart for WORDS



APPENDIX B

The CLOSE Project

CLOSE is an interactive program written in AED designed to help you learn basic accounting techniques. Specifically, CLOSE is designed to help you learn the following three aspects of accounting:

1) the process of posting entries to a ledger,

2) making adjustments to the ledger accounts at the end of a period, and

3) closing out the ledger acccunts.

CLOSE employs the technique of presenting several example transactions and then asking you to give the answers to several similar transactions. The program assumes that you will abide by certain conventions; these conventions are listed below:

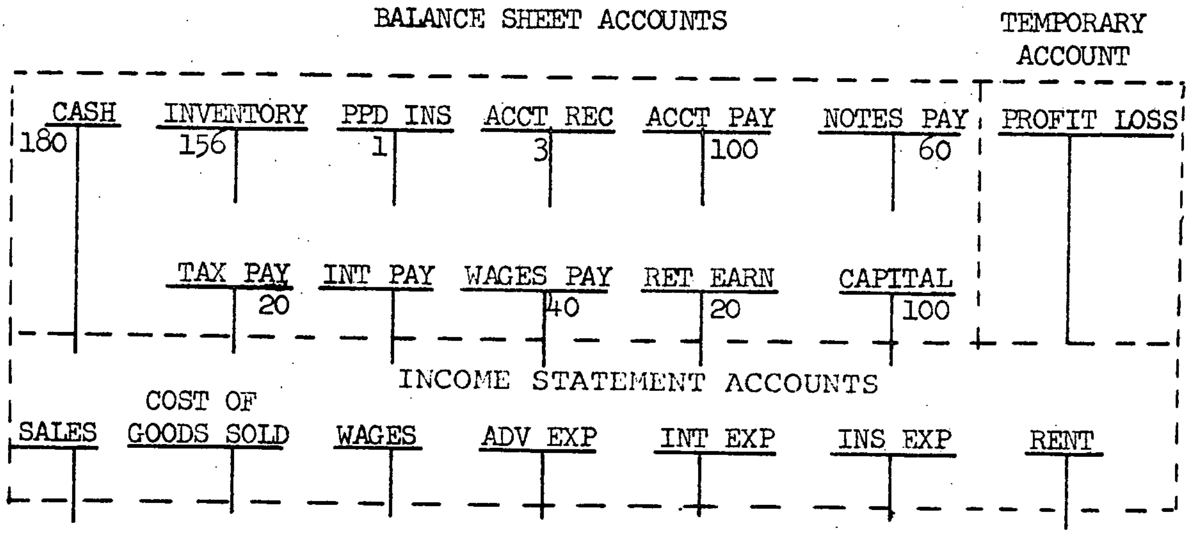
1) When "READY" appears on the screen, it means that the program is waiting for you. When you are ready to continue, and have typed a response, if one was expected, you should hit "new line".

2) Example transactions are denoted by the prefix "EXAMPLE". You are not expected to anser these, but you should make sure that you answer each one before you continue with the program.

3) Transactions which expect you to respond are prefixed by "QUESTION". The format of a response is similar to that of a journal entry, i.e., an account and the amount to be debited is specified, followed by the account and the amount to be credited. Specifically, you respond to a question by typing "dr" followed by the account to be debited, followed by the amount (in thousands of dollars); on the same line you then type "cr" followed by the account and the amount to be credited. Then hit "new line".

Figure 53 shows the T-accounts used in CLOSE as well as an initial balance sheet and income statement. Note that the accounts are laid out in an orderly manner. In gener, the scheme is first asset accounts, then liability accounts, and finally, income and expense accounts.

Figure 53



These accounts are the "books" of GEM, Inc. GEM is engaged in the manufacture and sale of Adzes, an esoteric plant made from a hemp plant and widely held to be the cure for all of man's ills.

You have been hired as an accountant and are responsible for keeping GEM'S books.

BALANCE SHEET

ASSETS		
cash	180	
inventory	156	
ppd ins	1	
acct rec	3	
TOTAL ASSETS		340
EQUITIES		
LIABILITIES		
acct pay	100	
notes pay	60	
tax pay	20	
int pay	0	
wages pay	40	
TOTAL LIABILITIES		220
ret earn	100	
capital	20	
TOTAL EQUITIES		120
		340

INCOME STATEMENT

sales	0	
cost of goods sold	0	
GROSS PROFIT		0
OPERATING EXPENSES		
wages	0	
adv exp	0	
int exp	0	
ins exp	0	
rent	0	
TOTAL OPERATING EXPENSES		0
NET PROFIT		0

<u>Using the ARDS Computer Terminal</u>

To communicate with the accounting tutor, the student must follow some general guidelines in operating the terminal:

1) All input to the computer is sent in "bursts" triggered by depressing the ""NEW LINE" key; each and every response must be followed by a "NEW LINE".

2) The computer is only "listening" to the terminal when the green "proceed" light is on; anything typed on the keyboard while the "wait" light is on will be ignored.

3) When the screen is full, a small "blob" will appear at the bottom of the screen; the white "ERASE" key <u>on the keyboard</u> will erase the screen and also notify the computer that it may continue to write. This "ERASE" key may be used to clear the screen at any time the green "proceed" light is on; one does not have to wait until the "blob" appears. However, CLOSE monitors the number of lines

on the screen, and it is unlikely that the student will have to use the "ERASE" key.

Since typing mistakes during input are bound to occur, two special characters are available for error correcting:

1) The commercial at (@) typed after a group of characters will deleteyped after a group of characters will delete all the chacters; whatever follows the commercial at sign will be interpreted as a new line, although the characters must be typed on the same line. Thus, 'this part will be deleted@this part will not'. A commercial at at always has this deleting property; consequently, one must be careful not to use it for any other purpose.

2) A number sign (#) typed after any character (or space) will delete that character. This may be done repeatedly to delete more that one character. For example, 'abcdef#' will be interpreted as 'abcde', while 'abcdef###' will be interpreted as 'abc'.

Each answer from the student to the tutor should not exceed one line in length; continuaticn lines are not allowed in the present configuration. Either upper cr lower case letters of the alphabet are acceptable. Non-alphabetic characters (other than numbers or the two correction characters just mentioned) should not be used.

The CLOSE program is adeveloping instructicnal program. In order to continue its improvement, we solicit your comments. These comments may be entered at any time after CLOSE has been started by typing an asterisk(*) followed by your comment. The CLOSE program enters these lines onto a disc file where they are saved for us to read at a later time.

Starting the Tutor

After an active line to the computer has been established, one should type 'r close 1', following it, as always, by a "NEW LINE". The computer will respond with 'W xxxx.x'. This loads the program. The tutor will start by requesting the student's name. Normally, a student will type in his full name, then depress "NEW IINE".

Control Words

CLOSE recognizes a number of control words which will allow the student to interact more effectively with the system. The commands are described below:

1) *: In order to provide comment to the system developers, a student may enter a line started with an asterisk. This line may be entered at any time and does not change the status of the program.

2) STOP: This command allows the student to quit the program before it has finished. "STOP", not an interrupt, should always be used to terminate the program as CLOSE modifies the display screen parameters and these must be reset upon termination of the program.

3) DISPLAY: In the event that the screen is inadvertently erased and the T-accounts lost, the "DISPLAY" command will redraw the T-accounts.

4) QUESTION: This command will retype the current question if it is erased.

5) SKIP [1]: In order to allow the student to proceed at his own pace, "SKIP" will allow the student to skip over questions or sections which he already understands. "SKIP 1" will skip to the next question. "SKIP" requests the student to indicate whether he wants to skip to the next question or to the next section.

Commands 1-5 may be typed whenever "READY" appears on the screen. There is an additional command which is only recognized after a question has been asked. This command is described below:

6) HELP: The student, in going through the program, may encounter a question which he cannot answer. The command "HELP" will give the student help in answering the question. CLOSE will accept a partial answer, i.e., "dr adv exp 10 (NEW LINE)", and will give feedback as to whether the partial anser is correct or not. Further, if a response is only

partially incorrect, the student need only type in the corrected section of his response. He need not retype his entire response.

Miscellaneous

1) If your answer is incorrect, CLOSE brackets the incorrect part with asterisks, i.e.

your answer: dr wrong acct 4

CLOSE's response: dr **wrong acct** 4

BIBLIOGRAPHY

1. Anderson, B.J., The Development of a Computer-Assisted Accounting Tutor", unpublished Bachelor's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, February 1970.

2. Anthony, R.N., Management Accounting: Text and Cases, Richard D. Irwin, Inc., Homewood, Illinois, 1964.

3. Bitzer, D.L. et al., "The PLATO System: Current Research and Developments", IEEE Transactions on Human Factors in Electronics, Vol. HFE-8, No. 2, June 1967, pp. 64-70.

4. Bitzer, D.L. and Suchman, "REPLAB -- A study in Scientific Inquiry Using the PLATO System", University of Illinois, Report R-260, 1965.

5. Bitzer, D.L., et al., "The Uses of PLATO: A Computer Controlled Teaching System", University of Illinois, Report R-268, 1965.

6. Braunfeld, P.G., "Problems and Prospects of Teaching with a Computer", Journal of Educational Psychology, Vol. 55, No. 4, 1964, pp. 201-211.

7. Bushnell, D.D., "Computer-Based Teaching Machines", Journal of Educational Research, Vol. 55, No. 9, June-July 1962, pp. 528-531.

8. Coulson, J.E. and Silberman, H.F., "Automated Teaching and Individual Differences", Audio-Visual Communications Review, Vol. 9, 1961, pp. 5-15.

9. Coulson, J.E., "A Computer-Based Laboratory For Research and Development in Education", in Coulson(11).

10. Coulson, J.E., "Effects of Branching in a Computer Controlled Autc-Instructional Device", _Journal of Applied Psychology_, Vol. 46, No. 6, 1962, pp. 389-392.

11. Coulson, J.E., _Programmed Learning and Computer-Based Instruction_, John Wiley & Sons, New York, 1962, pp. 58-66.

12. Cram, David, _Explaining "Teaching Machines" and Programming_, Fearon Publishers, Inc., San Francisco, California, 1961.

13. Crisman, D.A. (ed.), _Compatible Time Sharing System, A Programmer's Guide_, The M.I.T. Press, Cambridge, Massachusetts, 1965.

14. Crowder, N.A., "Automatic Tutoring By Intrinsic Programming", in Lumsdaine(29).

15. Crowder, N.A., "Intrinsic and Extrinsic Programming", in Coulson(11), pp. 58-66.

16. Crowder, N.A., "Intrinsic Programming: Facts, Fallicies, and Future", in Filep, R.T., _Prospectives in Programming_, The MacMillan Company, New York, 1963, pp. 84-115.

17. Deterline, W.A., _An Introduction to Programmed Instruction_, Prentice Hall, Edgewood Cliffs, New Jersey, 1962.

18. Dixon, G.T., "Analyzing Student Performance To Improve An Interactive Teaching Program", unpublished Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1970.

19. Englund, D.E. and Estavan, D.P., "CLASS -- The Automated Classroom", _Computer Applications 1961_ -

Prceedings of the 19961 Comruter Applications Symposium, R.S. Hollitch and B. Mittman (eds.), The MacMillan Company, New York, 1961, pp. 177-188.

20. Feingold, S., "A Flexible Language For Programming Computer/Human Interaction", System Development Corporation, Santa Monica, California, 1966.

21. Feingold, S.L. and Frye, C.H., "User's Guide To PLANIT", Report TM-3055/000/00, System Development Corporation, Santa Monica, California.

22. Fry, E., _Teaching Machines and Programmed Instruction_, McGraw-Hill Book Company, Inc., New York, 1963.

23. Garner, W.L., _Programmed Instruction_, The Center for Applied Research in Education, Inc., New York, 1966.

24. Grubb, R.E. and Selfridge, L.D., "Computer Tutoring in Statistics", _Computers and Auycmation_, Vol. 13, No. 3, March 1964.

25. Grubb, R.E., "Learner-Controlled Statistics", _Prcgrammed Learning_, January 1968, pp. 38-42.

26. Gruhl, F. "Development of a Semantic Memory for a Teaching Program", unpublished Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1968.

27. LEDGER, undocumented CAI accounting tutor developed by J. Macko for the ALP Project, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1969.

28. Licklider, J.C.R., "Preliminary Experiments in Computer-Aided Teaching", in Coulson(11), pp.

217-239.

29. Lumsdaine, A.A. and Glaser, R., _Teaching Machines and Programmed Learning_, National Education Association, Washington, D.C., 1960.

30. MacAims Programming Guide, Project MAC, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1969.

31. Nordman, B.J., "Teaching Machines and Programmed Instruction: An Introduction and Overview", Report No. 260, University of Illinois, 1968.

32. Pressey, S.L., "Apparatus Which Gives Tests and Scores", _School and Society_, Vol. 23, 1926, pp. 373-376.

33. Pressey, S.L., "A Machine for Automatic Teaching of Drill Material", _School and Society_, Vol. 25, 1927, pp. 549-552.

34. Pressey, S. L., "A Third and Fourth Contribution Toward the Coming 'Industrial Revolution' in Education", _School and Society_, Vol. 36, 1932, pp. 668-672.

35. Rath, G.J., "The Development of Computer-Assisted Instruction", _IEEE Transactions on Human Factors in Electronics_, Vol. HFE-8, No. 2, June 1967, pp. 60-63.

36. Rath, G., Anderson, N.S., and Brainerd, R.C., "The IBM Research Center Teaching Machine Project", in Galanter, E., _Automatic Teaching: The State of the Art_, John Wiley & Sons, New York, 1959, pp. 117-130.

37. Rockart, J.F., Scott Morton, M.S., and Zannetos, Z.S., "Associative Learning Project -- Phase 1 System", unpublished working paper (440-70), Sloan School

of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1970.

38. Schiff, M.A., "The Planning and Evaluation of a Computer-Assisted Programmed-Instruction Algorithm", unpublished Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1969.

39. Scott Morton, M.S., and Zannetos, Z.S., "Efforts Toward an Associative Learning Instructional System", Proceedings of the IFIP Congress 68, Edinburgh, Scotland, August 1968. (Also MIT Working Paper No. 355-68).

40. Simmons, R.F., "Natural Language Question-Answering Systems: 1969", Communications of the ACM, Vol. 13, No. 1, January 1970, pp. 15-30.

41. Skinner, B.F., "The Science of Learning and the Art of Teaching", Harvard Educational Review, Vol. 24, 1954, pp. 86-97.

42. Skinner, B.F., "Teaching Machines", Scientific American, November 1961.

43. Skinner, B.F. and Holland, J.G., "The Use of Teaching Machines in College Instruction", in Lumsdaine (29).

44. Spolsky, B., "Some Problems of Computer-Based Instruction", Behavioral Science, November 1966, pp. 487-496.

45. Stotz, R.H. and Cheek, T.B., "A Low-Cost Graphic Display for a Computer Time-Sharing Console", 8th National Symposium on Information Display, May 1967.

46. Stotz, R.H., "A New Display Terminal", Computer Design, April 1968.

47. Uttal, W.R., Pasich, T., Rogers, M., and Hieronymous, R., "Generative Computer Assisted Instruction", Communication No. 243, Mental Health Research Institute, University of Michigan, 1969.

48. Uttal, W.R., "On Conversational Interaction", in Coulson(11), pp. 171-190.

49. Ward, J.E., "Graphic Output Performance of the ARDS Terminal with the Tektronix Type 611 Storage Monitor", Project MAC Memorandum MAC-M-368, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1968.

50. Weizenbaum, J., "ELIZA, A Computer Program for the Study of Natural Language Communications Between Man and Machine", Communications of the ACM, Vol. 9, No. 1, January 1966, pp. 36-49.

51. Zannetos, Z.S., "Programmed Instruction and Computer Technology", The Accounting Review, Vol. XLII, No. 3, July 1967.